TRABAJO FIN DE CARRERA

TÍTULO: Diseño de una aplicación de localización en redes de sensores con protocolo IEEE 802.15.4

AUTOR: David Aguallo Nicolás

DIRECTOR: Carles Gómez Montenegro

FECHA: 28 de junio de 2005

Título: Diseño de una aplicación de localización en redes de sensores con protocolo IEE 802.15.4

Autor: David Aguallo Nicolás

Director: Carles Gómez Montenegro

Fecha: 28 de junio de 2005

Resumen

En este proyecto se explican las pautas que se han seguido para crear y validar una aplicación de localización en redes de sensores.

Puede que el título del proyecto desconcierte al lector, ya que en realidad, el hecho de que los sensores utilicen como medio de transmisión radio el protocolo IEEE 802.15.4 (ZigBee) no influye en los resultados del proyecto.

El estándar ZigBee ha sido creado casi específicamente para estos dispositivos empotrados debido a la necesidad de minimizar el consumo, y aunque la velocidad de transmisión sea pequeña (250kbps) es suficiente para una red de sensores.

Para el proyecto se ha dejado un poco apartado el estándar Zigbee (aunque se utiliza para el envio de mensajes entre los nodos de la red) para centrarse en el objetivo principal, que es la localización de un nodo en la red utilizando la tecnología disponible en cada dispositivo.

Para realizarlo, se ha estudiado la viabilidad de utilizar el tiempo de vuelo de una señal sonora para calcular la distancia a un punto conocido del espacio (otro dispositivo), y una vez encontrado un modelo válido con un error inferior a 5 cm, se ha implementado una aplicación en un pc que localice el dispositivo en 2 dimesiones utlizando las medidas conseguidas mediante la técnica anterior y un método de localización llamado triangulación de distancias.

El sistema se ha validado y se ha comprobado la eficacia del método y sus limitaciones.

Title: Design of a localization application in nets of sensors with protocol IEE 802.15.4
Author: David Aguallo Nicolás
Director: Carles Gómez Montenegro
Date: June, 28th 2005

Overview

This project explains the rules that have been continued to create and validate a localization application in sensor's network.

Maybe the title of the project confuses the reader, because of the fact that the sensors uses the protocol IEEE 802.15.4 (ZigBee) like medium of transmission does not influence in the results of the project.

The standard ZigBee has been created specifically for these embedded devices due to the necessity of minimize the consumption, and although the transmission speed is small (250kbps) it is enough for a sensor's network.

For the project it has been left the standard Zigbee (although it is used for the exchange of messages among the nodes of the net) to be centered in the main objective that is the localization of a node in the network using the available technology in each device.

To carry out it, it has been studied the viability of using the time of flight of a sound to calculate the distance to a well-known point of the space (another device), and once chosen a valid model with an error smaller than 5 cm., an application has been implemented in a pc that locates the device in 2 dimensions using the measures capturated by the previous technique and a method of localization called triangulation of distances.

The system has been validated and it has been proven the effectiveness of the method and its limitations.

ÍNDICE

INTRODUCCIÓN

Motivación

El nuevo empuje de las redes de sensores y sobre todo el hecho de que se constituyan sin infraestructura previa, hace cada vez más interesante y en algunos casos necesaria la localización de los participantes de la red.

Cada vez los nodos de la red son más potentes y cada vez más autónomos. Gracias a los protocolos ad-hoc, podemos fácilmente introducir nuevos participantes en la red simplemente situándolos en la zona de cobertura de otro participante.

El conocimiento de la posición de un nodo seria muy fácil si a cada uno le introdujéramos tecnología para calcular la posición mediante GPS, pero si nos paramos a pensar en la cantidad de nodos que puede albergar una red de sensores y el coste de la tecnología por unidad, nos daríamos cuenta que es demasiado elevado. De esta premisa parte la idea de localizar a los nodos de una red utilizando como espacio conocido la propia red y la propia electrónica de los sensores, ya que si se consigue un modelo lo bastante fiable, se conseguiría el mismo fin sin introducir más gastos en el sistema.

Objetivos

El objetivo de este trabajo persigue conseguir la localización de los nodos de una red de sensores utilizando la tecnología disponible en cada nodo.

Se ha estudiado y probado la posibilidad de localización mediante la técnica denominada “tiempo de vuelo de una señal”, utilizando en este caso una señal sonora, producida por un altavoz y recogida por un micrófono, ambos equipados en los sensores con los cuales se ha realizado el trabajo.

La técnica de medida del tiempo de vuelo se utiliza para conocer la distancia a un nodo de referencia. Para conocer la posición del nodo, se utiliza triangulación a partir de 3 nodos.

Estructura del documento

Este documento esta estructurado en 9 capítulos. En el capítulo 2 se intenta explicar la situación actual de las redes de sensores y como se localizan actualmente los participantes de la red. El capítulo 3 presenta los sensores que se han utilizado en este proyecto. El 4 y el 5 muestran de forma general como se piensa implementar el sistema. El 6 y el 7 reflejan las pruebas realizadas para validar el sistema y finalmente se hace alusión al consumo de batería y a las implicaciones medioambientales.

CAPITULO 1: ESTADO DEL ARTE

En la actualidad, las investigaciones sobre localización de dispositivos que pertenecen a una red es una de las vías de estudio importantes en el mundo de las telecomunicaciones.

Existen muchos métodos válidos, dependiendo de las condiciones en que se forme la red, su extensión, etc. por ejemplo, GPS y GSM proporcionan la localización de participantes de sus redes, con un error de metros, utilizando radiofrecuencia, pero no es posible su utilización en interiores.
La ventaja de estas soluciones es que la señal utilizada para la localización tiene un gran alcance y con pocos dispositivos se puede cubrir una gran extensión de terreno.

Sin embargo existen otras soluciones a menor escala, que normalmente utilizan ultrasonidos, ya que el error que se suele cometer es de apenas 2 o 3 cm., o sonido audible, pero que por su alcance se utilizan en interiores, y es necesario un elevado numero de nodos para cubrir, por ejemplo, una vivienda.

Otras soluciones menos utilizadas se basan óptica, básicamente en infrarrojos y láser, que se destinan casi exclusivamente a la medida de distancias sin el fin de localizar un objeto.

Todos los métodos tienen sus ventajas y sus inconvenientes.
Para localización en exteriores, el mas utilizado en la actualidad es GPS, pero para interiores no existe todavía ningún sistema predominante, que cumpla de forma satisfactoria las necesidades de consumo, precisión y robustez ante errores, y aunque los resultados nos llevan a pensar en soluciones basadas en ultrasonidos hay sistemas que actualmente están en estudio que se basan en tecnologías de uso común y aprovechan que ya están desplegadas para utilizarlas como medio de localización, como Bluetooth, UWB o telefonía móvil en general.

El método mas utilizado para calcular la posición de un nodo de la red es la triangulación de distancias medidas desde diferentes puntos conocidos a priori. La triangulación se puede basar en lateración (utilizando las distancias para localizar el nodo) o en angulación (utilizando ángulos).

Para el cálculo de las distancias, los métodos más utilizados son mediante la atenuación de una señal o bien mediante el tiempo de vuelo de una señal. También existen métodos que utilizan visiones artificiales con percepción de profundidad, pero son demasiado costosos y difíciles de implementar en sistemas empotrados.

En este proyecto se pretende utilizar la técnica de triangulación por lateración, utilizando el tiempo de vuelo de una señal sonora para localizar un nodo en 2 dimensiones.

CAPITULO 2: PLATAFORMA DE DESARROLLO

2.1. Kit de CrossBow

Para la realización del proyecto se ha utilizado el kit “MOTE-KIT2400” (http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Kit_Datasheet.pdf) fabricado y distribuido por CrossBow (www.xbow.com).

El kit se compone de 8 motes tipo micaZ, 7 placas con sensores, 1 programador serie, un programador Ethernet y una placa de adquisición de datos.

En concreto, se han utilizado los motes, las placas sensoras y el programador serie

2.1.1. Programador MIB-510

Se conecta al PC a través del puerto serie. Gracias a esta conexión es posible tanto la programación de los micaZ como la transmisión de información hacia el PC.

2.1.2. MicaZ

Son las plataformas sobre las cuales se insertan las placas sensoras. Están dotadas de procesador, memoria flash y chipset ZigBee para comunicación radio.

Llevan incorporado un conector tanto para las placas sensoras como para el programador, así como 3 leds como dispositivos de salida controlados.

2.1.3. Placas Sensoras

El kit trae 2 tipos de placas sensoras:
Las placas MTS300 incorporan sensor de luz, sensor de temperatura, altavoz y micrófono.
Las placas MTS310 son mas completas, incorporan además sensor magnético y de aceleración.
Para este proyecto se han utilizado ambas placas, ya que los dos modelos están dotados con micrófono y altavoz.

2.2. TinyOS

TinyOS es el sistema operativo que utilizan los motes de CrossBow.
Es un SO especialmente diseñado para sensores, por lo que es liviano y su código se ejecuta muy rápido.
Cuando se carga una aplicación en los sensores, ésta se compila junto al sistema operativo y se carga en el sensor, por los que no es necesaria una instalación previa del TinyOS en los motes.
Dispone de librerías para el manejo de los componentes del mote.

2.3. NesC

NesC es el código utilizado para programar las aplicaciones que se ejecutaran en los sensores y con el que está escrito TinyOS y sus librerías.

NesC es una variación de C con algunas limitaciones, creado específicamente para dispositivos empotrados (Network Embedded System C) basado en tareas y orientado a eventos.

CAPITULO 3: APLICACIÓN DE LOCALIZACIÓN

La técnica que se ha utilizado en este proyecto para localizar la posición del elemento deseado es la triangulación, utilizando nodos con posición conocida como referencias del sistema, y para la adquisición de las distancias a los nodos de referencia se ha utilizado la técnica del tiempo de vuelo de una señal sonora.

3.1. Triangulación

Suponiendo tres puntos en el plano cuyas posiciones son conocidas, podemos conocer la posición de un cuarto nodo mediante las distancias a cada uno de los tres nodos de referencia.

Es el mismo método que se utiliza en la localización mediante GPS, la diferencia es que en este proyecto se han utilzazo 2 dimensiones en lugar de 3.

Se utilizan las distancias a cada nodo como radio para calcular una circunferencia. La intersección de 2 circunferencias da como resultado 2 puntos en el plano, de los cuales, descartaremos uno gracias a la circunferencia calculada a partir del tercer nodo.

El método queda mas claro en la Figura 3.1.1.

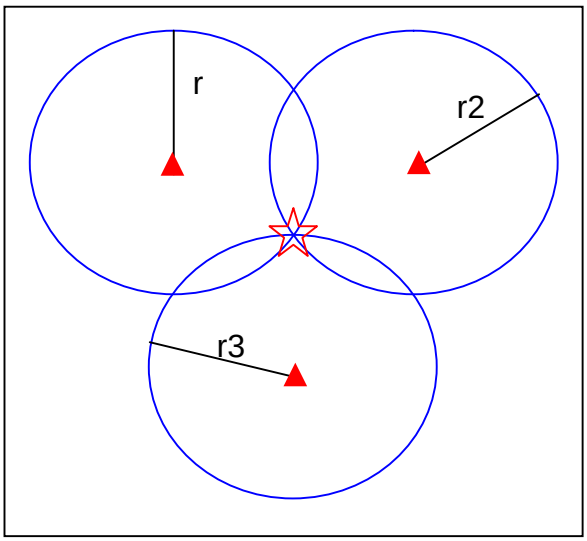


Fig. 3.1.1 Ejemplo de triangulación a partir de 3 distancias.

Para calcular la posición y mostrar el conjunto del sistema gráficamente, se ha programado una aplicación en C# en el PC.

3.2. Adquisición de distancias

Como ya se ha dicho, para la adquisición de distancias se ha utilizado el tiempo de vuelo de un tono emitido por el altavoz de un mote.

Todos los nodos de referencia utilizan la misma aplicación, el diagrama utilizado es extensible a todos los nodos de referencia que se incluyan en el sistema (Fig.3.2.1)

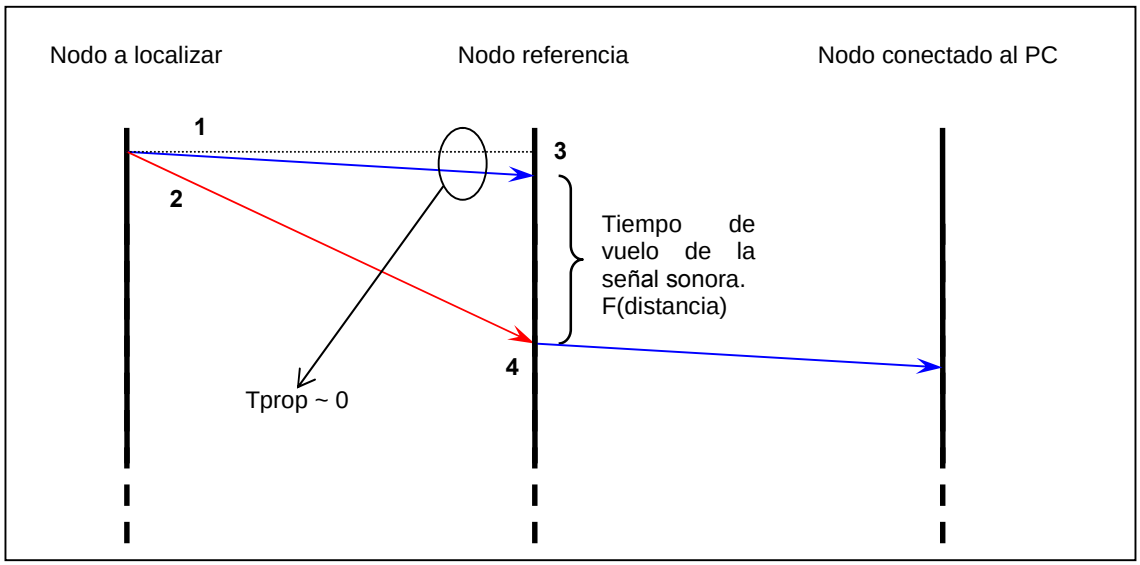


Fig. 3.2.1 Pasos necesarios para la adquisición de la distancia

Descripción de los pasos

1.- El nodo a localizar envía una señal radio (ZigBee) dirigida a los nodos con posición conocida. Esta señal es utilizada como sincronización (Fig. 3.2.1 punto 1).

2.- Inmediatamente después de enviar la señal radio, emite un todo por el altavoz a 4,3 Khz., de duración 0,1 s (Fig. 3.2.1 punto 2).

3.- Los nodos de referencia reciben el mensaje radio y guardan el valor del temporizador en una variable local (Fig. 3.2.1 punto 3).

La velocidad de propagación de la señal radio es prácticamente la de la luz, por lo que se puede suponer que el instante de recepción del mensaje radio es el mismo que el instante de envío de la señal sonora.

4.- Los nodos de referencia reciben la señal sonora mas tarde que la señal radio, ya que implícitamente viaja a la velocidad del sonido, muy

por debajo de la velocidad de la señal radio, por lo que es un tiempo suficiente como para poder medirlo.
Los nodos guardan el valor del temporizador en otra variable local (Fig. 3.2.1 punto 4).

5.- Mediante la diferencia de los valores guardados se obtiene el valor del tiempo de vuelo de la señal sonora, y se envía al nodo recolector conectado al PC, que se encargará de procesar el dato (Fig. 3.2.1 punto 5).

CAPITULO 4: IMPLEMENTACIÓN

4.1. Esquema general

De forma general, el sistema se basa en un nodo en posición desconocida, que se desea localizar, 3 nodos de referencia, cuya posición es conocida y un nodo conectado a un PC, que retransmitirá los paquetes radio hacia el ordenador.

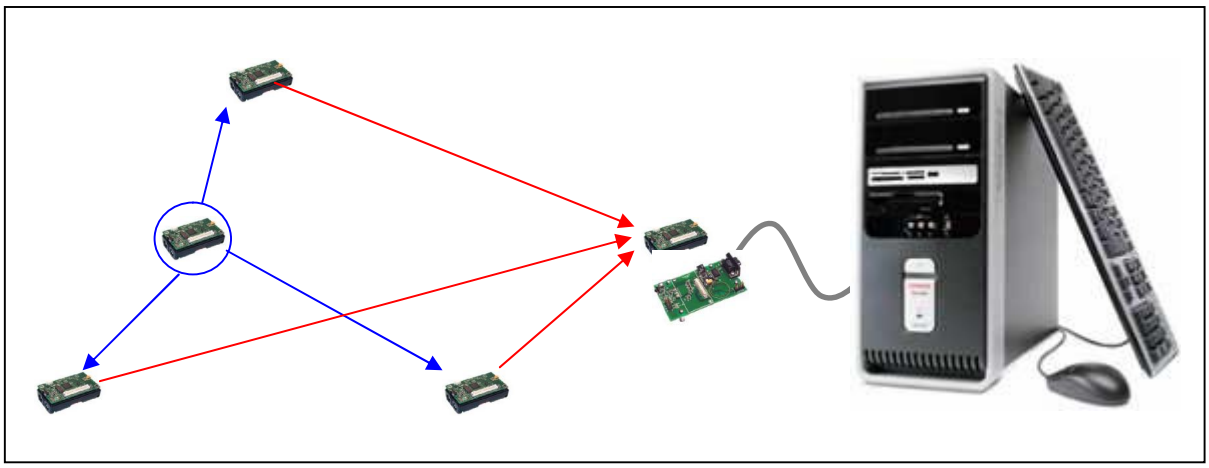
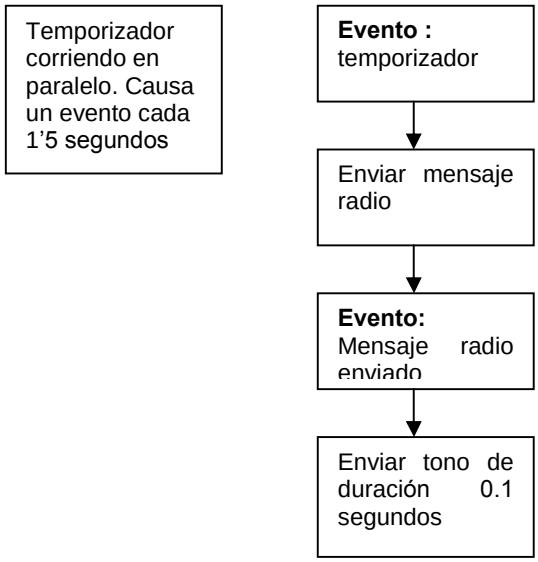


Fig. 4.1.1 Esquema general del escenario de pruebas.

4.2. Código utilizado

El código utilizado tanto por la aplicación como por los sensores esta disponible en el anexo de este proyecto.

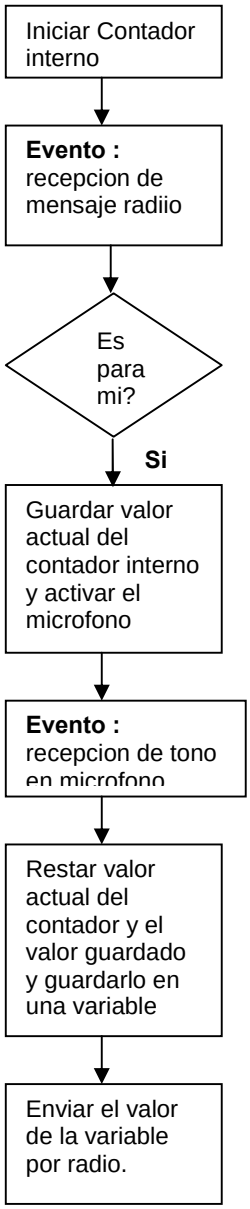
4.2.1. Diagrama del emisor de tonos.



Como ya se ve en el diagrama, y se ha comentado antes, TinyOS y NesC están orientados a eventos, por lo que no es necesario preguntar continuamente por el valor del temporizador.

El segundo evento se llama "sendDone" y lo genera TinyOS cada vez que se envía un paquete mediante radio. Es muy útil en casos como este, para ejecutar código justo después de enviar un paquete.

4.2.2. Diagrama del receptor de Tonos.



El contador interno utilizado se llama “SysTime”, es el mas preciso que se ha encontrado en las librerías de TinyOS.
Se trata de un contador de 32 bits que funciona a una frecuencia de 921.6 Khz., lo que da una resolución de prácticamente 1 microsegundo (1 tick = 1.085 microsegundos).

De todas formas, para acarrear menos error, todos los cálculos de este proyecto se ha realizado utilizando ticks en lugar de segundos.

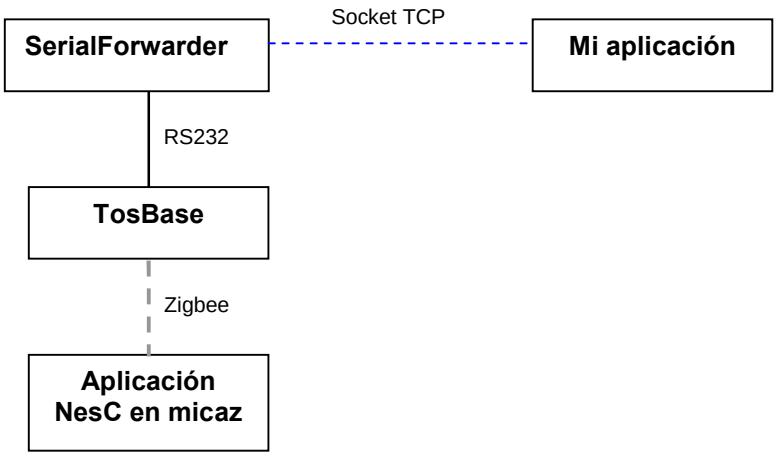
4.2.3. Nodo recolector.

El nodo recolector es el que se conecta al PC mediante la placa de programación MIB510.

En este nodo se carga una aplicación llamada “TosBase” cuyo cometido es simplemente transferir los paquetes que llegan vía radio hacia el puerto serie (RS232) conectado al PC.

Esta aplicación viene suministrada por TinyOS en un apartado de aplicaciones estándar.

Para que el PC sea capaz de recoger los datos, se ejecuta una aplicación proporcionada por CrossBow, denominada “SerialForwarder”, que se encarga de recoger los paquetes que llegan por el puerto serie y los reenvía por un socket TCP. De esta forma, si programamos una aplicación capaz de leer y escribir en un socket TCP, podremos comunicarnos con los nodos micaZ.



4.2.4. Código del PC.

La aplicación de localización se ha realizado utilizando lenguaje de programación C#.

Se ha procurado una interfaz amigable e intuitiva, en modo gráfico, que tras realizar su tarea, nos devuelve tanto la posición en coordenadas como la localización gráficamente.

La aplicación se conecta mediante sockets TCP a la aplicación SerialForwarder proporcionada por CrossBow (4.2.3 Nodo recolector).

Mediante esta conexión se reciben los bytes del paquete emitido por todos los nodos del sistema.

A partir de los bytes recogidos, la aplicación extrae el identificador de cada paquete para comprobar si proviene de un nodo de referencia. Si es así, se extraen los 4 bytes correspondientes al número de ticks del tiempo de vuelo y se le aplica la ecuación para calcular la distancia.

CAPITULO 5: EXPERIMENTACIÓN Y RESULTADOS

Una vez seguros de lo que se quiere implementar, se han hecho diferentes pruebas para escoger el mejor método de de resolución del proyecto.

Para el cálculo de los tiempos teóricos de vuelo de la señal sonora, se ha utilizado la velocidad de propagación del sonido en el aire (340 m/s).

Se empezó comprobando la calidad y naturaleza de las graficas que representan la distancia en función del tiempo de vuelo del tono emitido, que llamaremos filtrado del tiempo de vuelo. Una vez escogida una solución satisfactoria para la captura de los tiempos, se programó la aplicación que computa los tiempos y calcula la posición del nodo.

5.1. Filtrado del tiempo de vuelo

Lo primero que hacia falta acotar era el tiempo mínimo de duración del tono emitido para que el micrófono fuese capaz de detectarlo, además del tiempo mínimo entre todos para que la medida actual no fuera contaminada por la medida anterior.

La figura 5.1.1 muestra los resultados a una distancia de 1 metro, utilizando un tono de duración 1 segundo.

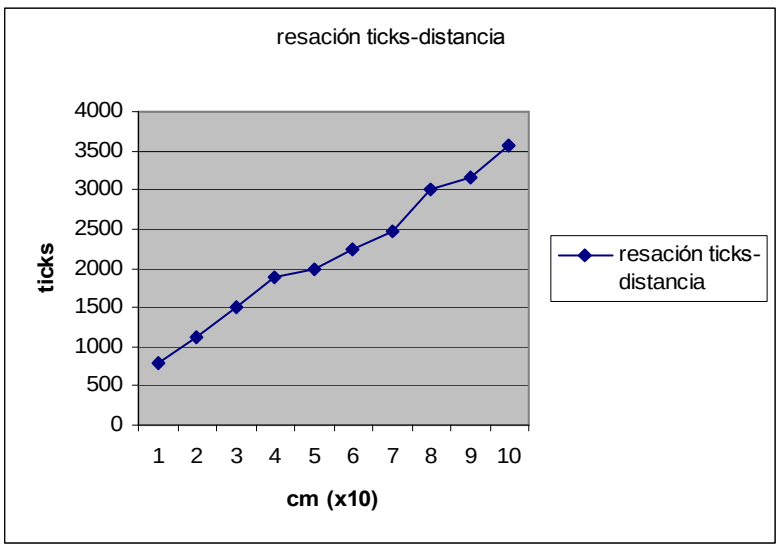


Fig. 5.1.1 Tono de duración 1 segundo

La figura 5.1.2 muestra el mismo resultado utilizando un tono de 0.1 segundos de duración.

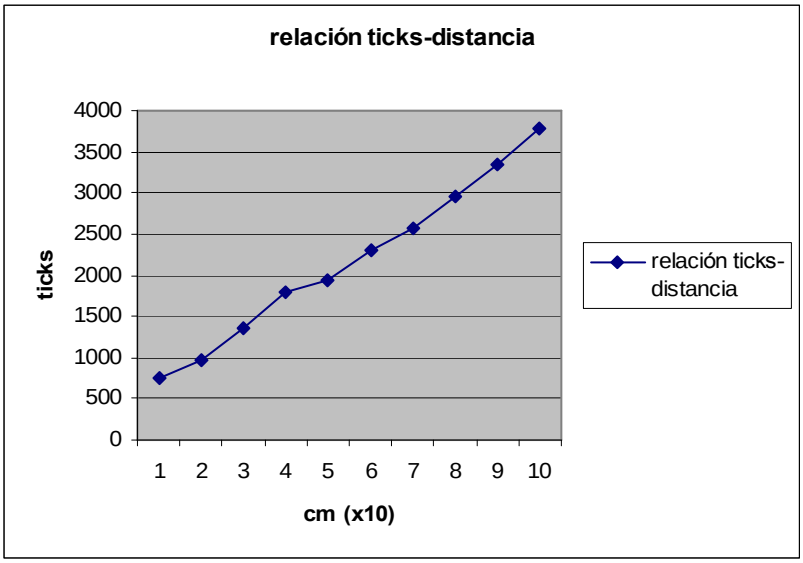


Fig. 5.1.2 Tono de duración 0.1 segundo

Estos son los márgenes de valores que se han utilizado para esta prueba (0.1 - 1 segundo). Las gráficas apenas muestran diferencias, sin tener en cuenta las variaciones de los valores debido al error implícito al tomar medidas utilizando una escala tan pequeña (microsegundos).
No se han probado tonos de duración mayor a 1 segundo, y por debajo de 0.1 segundos, el pitido es prácticamente inaudible al no tener tiempo de excitar el altavoz.

Se decide utilizar un tiempo de tono de 0,1 segundos, tanto para perturbar lo mínimo la calidad ambiental en el laboratorio, como para minimizar el consumo de batería.

El tiempo mínimo entre pitidos se ha tenido que estudiar utilizando el receptor.

Se han probado tiempos entre pitidos desde 0.1 segundos hasta 5 segundos.

Entre 5 segundos y 1 segundos, los resultados son idénticos (salvando otra vez el error de la medidas), pero para tiempos inferiores a 1 segundo, el micrófono todavía esta levemente alterado por el tono anterior, así que se decide dar un margen y utilizar 1.5 segundos entre pitidos.

5.1.1. Primera aproximación

Como primera aproximación, se realizó una aplicación que capturase un número limitado de tiempos y calculase la media de los tiempos. Para intentar hacer un primer filtrado de los tiempos, se elimina el tiempo máximo y el tiempo mínimo. La ganancia del micrófono se fijo a 64.

El resultado fue bastante prometedor, ya que la grafica se asemeja bastante a una recta, y la distancia máxima a la que se consiguieron resultados coherentes rondaba los 3 metros, con pequeñas imperfecciones que a primera vista son fáciles de solucionar (Fig. 5.1.1.1).

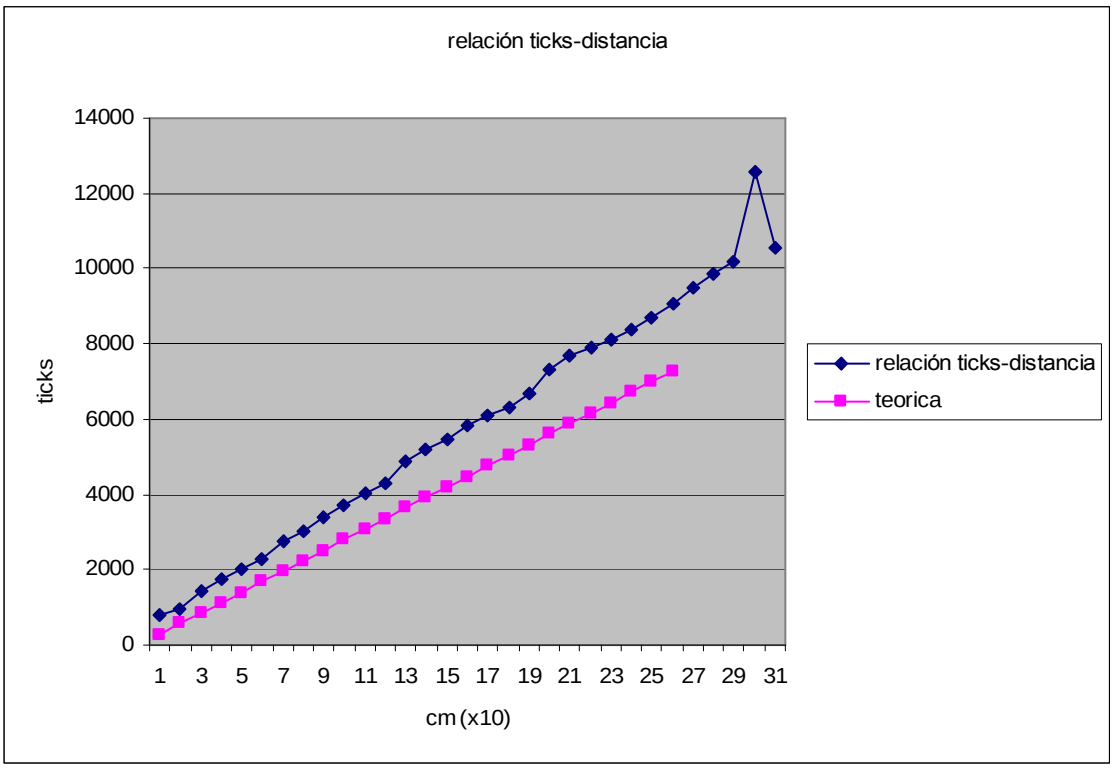


Fig. 5.1.1.1 grafica eliminando mínimo y máximo, ganancia 64

El problema es que estas medidas se hicieron enfocando los micrófonos y los altavoces. Las medidas son mejores en el caso de tener solo un emisor y un receptor, pero si suponemos el escenario real, no es posible que el emisor este enfocado hacia los 3 receptores (mínimo) a la vez, con lo cual se decidió repetir la experiencia enfocando emisor y receptor hacia arriba.

El resultado que se espera es una progresión de valores mas dispersa, con menos tendencia lineal, ya que al enfocar emisor y receptor hacia arriba, el sonido llega con mayor atenuación, cosa que afecta a la hora de captar el tono por parte del micrófono.

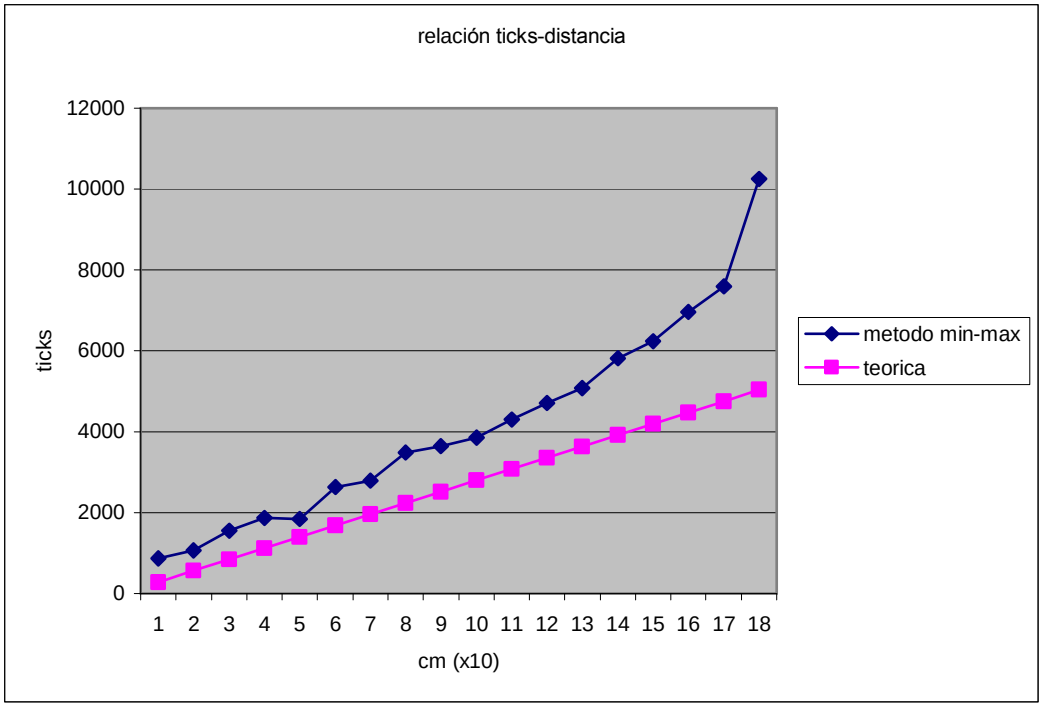


Fig. 5.1.1.2 Resultados enfocando emisor y receptor hacia arriba.

Como era de esperar, los resultados son algo peores en este caso. La grafica pierde linealidad, y la distancia máxima a la que el micrófono capta el sonido descienda a 2 metros aproximadamente.

Aunque se ha comprobado que los resultados son peores, el resto de la pruebas se han realizado enfocando altavoz y micrófono hacia arriba, ya que es una situación mas real.

5.1.2. Ganancia del micrófono

El siguiente paso que se escogió fue modificar la ganancia del micrófono para observar si los resultados mejoraban.

El micrófono se puede ajustar con una ganancia entre 0 y 256 (se utiliza un byte sin signo). Este valor decide el volumen de captación del micrófono.

Las graficas de las figuras 5.1.1, 5.1.2, 5.1.1.1 y 5.1.1.2 están realizadas con una ganancia de 64.

Las pruebas se realizaron cambiando la ganancia del micrófono de 32 en 32, iniciando en 32, ya que una ganancia de 0 desactiva el altavoz.

Tras realizar pruebas, se decide utilizar una ganancia de 128 (figura 5.1.2.1), que es la máxima ganancia en la que no se observan valores afectados por el ruido ambiente, como por ejemplo tiempo por debajo del teórico.

Se programa el nodo que capta el tono con una ganancia de micrófono de 128

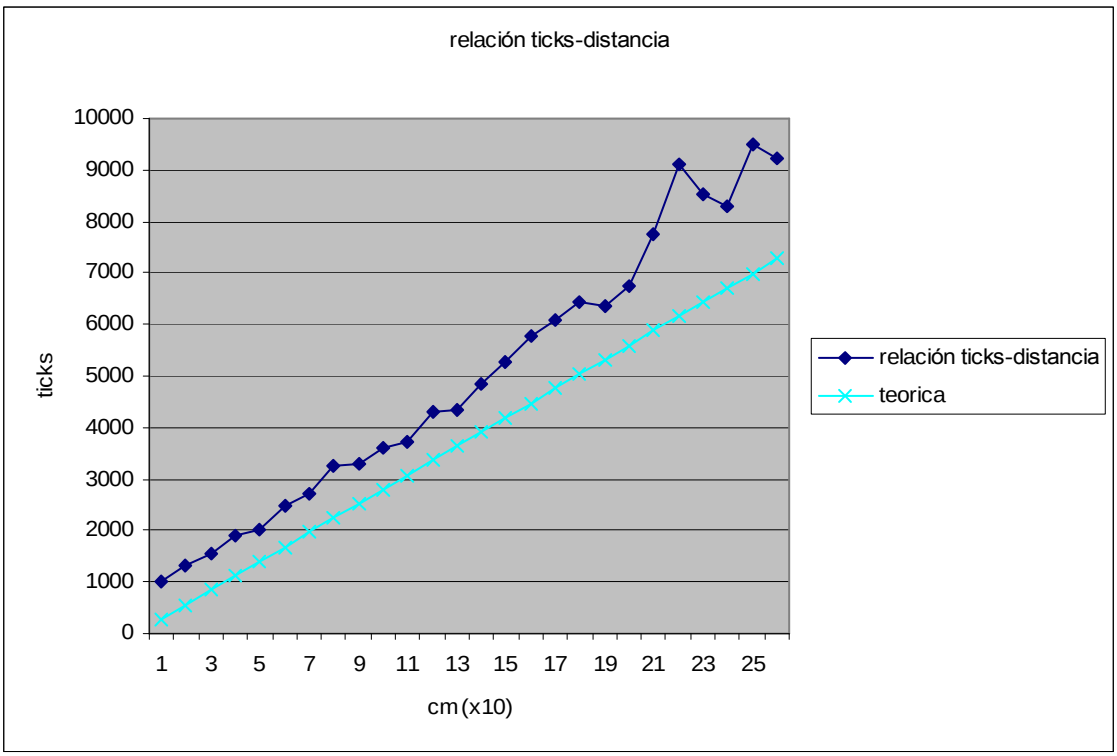


Fig. 5.1.2.1 Relación ticks-distancia, ganancia 128.

Se observa que al aumentar la ganancia del micrófono hemos conseguido que la grafica tenga menos tendencia exponencial, y pase a ser más lineal. Esto se debe a que el tiempo de detección de sonido es menor, y a distancias mayores la medida no se ve tan afectada.

Por otra parte, en esta gráfica se observa como a distancias mayores de 2 metros las medidas parecen incoherentes.

5.1.3. Algoritmo con umbral

Se ha implementado un algoritmo de filtrado de valores basado en un articulo[1] de la revista IEEE wireless communications.

En dicho algoritmo, existen dos valores umbral (máximo y mínimo). Los tiempos que se van capturando se comparan con los valores umbrales. Si el valor esta entre los umbrales, se guarda y los nuevos umbrales serán la media entre el umbral y el valor.

En el caso que el valor este fuera de los umbrales, se descarta el dato y los umbrales siguen en su sitio.

El valor final será la media de los umbrales cuando se encuentren lo suficientemente próximos.

[1] "A system of tracking and mapping in real environments" diciembre de 2005

El problema de este método es que los tiempos capturados tienen mucha dispersión, por lo que los umbrales van variando demasiado y nunca llegaban a estar suficientemente próximo.

Existe otro problema que es cuando primer valor que se captura es un valor contaminado por un ruido externo o muy desviado de la media teórica. Siendo así, si el valor se encuentra entre los umbrales, éstos se modificaran, de forma que muy probablemente la media teórica quede fuera de los umbrales y los valores correctos se ignoren siempre.

Para intentar solucionar el problema se reprogramó el algoritmo de la siguiente forma:

Si el valor esta entre los umbrales, se guarda y los nuevos umbrales serán la media entre el umbral y el valor.

En el caso que el valor este fuera de los umbrales, se descarta el dato, pero los umbrales se modifican de la misma forma que si el valor fuera correcto, el nuevo umbral será la media entre el umbral y el valor.

El valor final se calcula como la media de todos los valores que han caído entre los umbrales después de un número limitado de capturas.

Con este nuevo algoritmo se soluciona el problema de que los valores sean muy dispersos, pero solo para que la aplicación no entre en un bucle, ya que el final del proceso esta marcado por el usuario. Para conseguir resultados parecidos a los de los otros métodos hacían falta series de entre 20 y 30 capturas. Esto supone un tiempo de entre 30 y 45 segundos.

Para la figura 5.1.3.1 se han hecho 60 capturas

El resultado obtenido es mejor que utilizar el método de eliminación del valor mínimo y máximo, ya que se aproxima mas al valor teórico, y el protocolo hace al sistema robusto ante las variaciones que tienen los valores por el error de medida y ante ruidos externos que puedan afectar a las capturas.

La desventaja es que es necesario mucho más tiempo que los demás métodos probados, sobre todo porque cuando tenemos el umbral cerca del valor teórico, se eliminan muchos valores que llegan por encima del umbral.

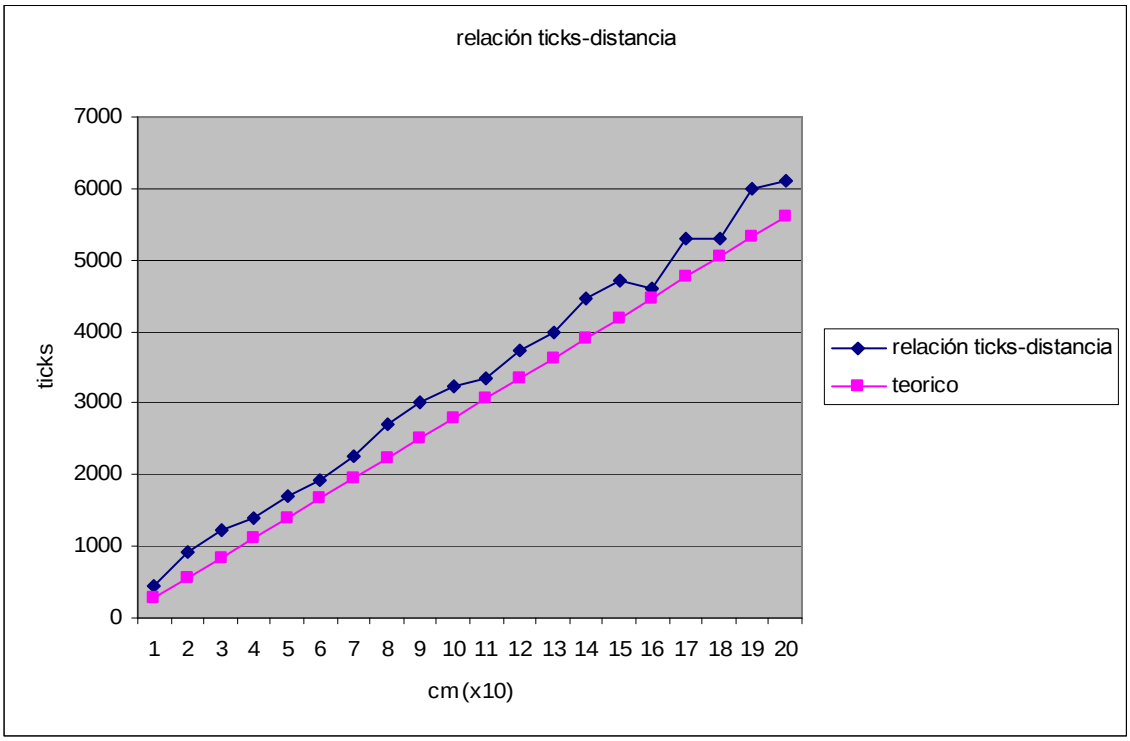


Fig. 5.1.3.1 Grafica resultado de utilizar un algoritmo con umbrales

En el artículo que ha servido como base de este método, se utilizan kits de ultrasonidos, cosa que hace pensar que el mismo algoritmo con dicho kit podría mostrarnos resultados mucho más satisfactorios que cualquier algoritmo con sonido audible.

5.1.4. Eliminar valores erróneos.

Examinando los resultados de las capturas de los valores, y después de la experiencia con el protocolo anterior donde se descartaban muchos valores por encima de los umbrales permitidos, se observa que los valores son bastante dispersos para una misma medida, pero nunca son inferiores a su valor teórico, por lo que se intentan métodos parecidos a la primera aproximación, pero eliminando mas valores probablemente erróneos.

Así la práctica es ir eliminando los valores de tiempo más grandes, con la finalidad de que la media de los valores restantes se aproxime más a la medida teórica.

En la figura 5.1.4.1 están representados resultados de eliminar los 6 valores máximos de los 10 recogidos, y el resultado de de eliminar los 7 valores máximos.

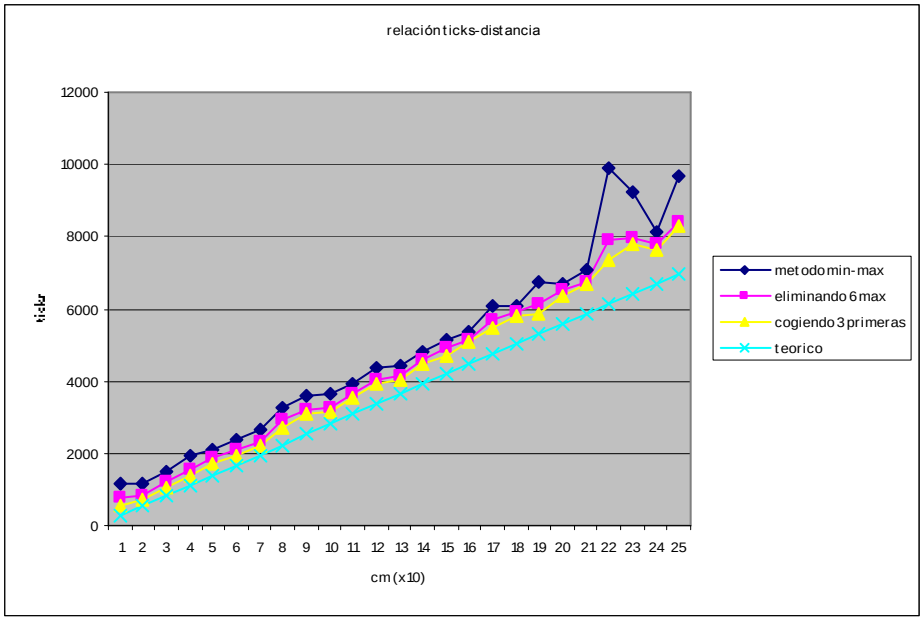


Fig. 5.1.4.1 Gráficas comparativas

Concordando con lo esperado, conforme vamos haciendo media con menos valores (es decir, nos quedamos con los mínimos), mas se aproxima la grafica a la función teórica.

Esto se puede ver también si dibujamos la progresión de los valores de menor a mayor.

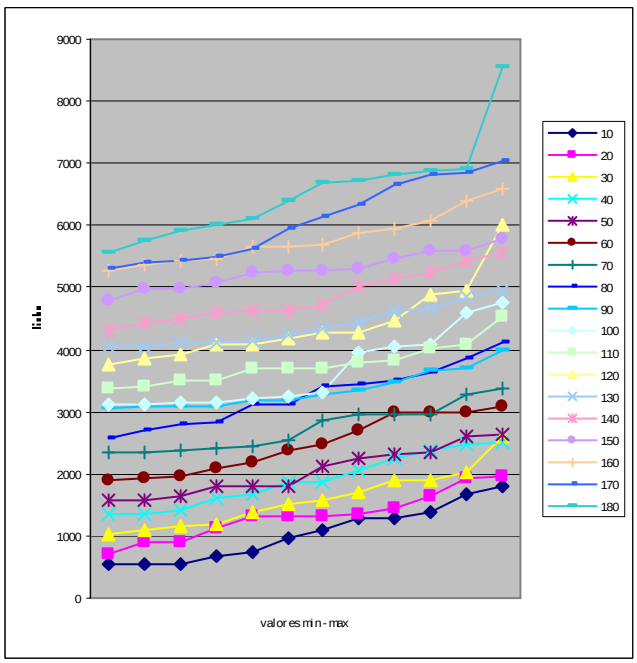


Fig. 5.1.4.2 Progresión de los valores ordenados de menor a mayor

5.1.5. Modelo utilizado

Llevando el modelo del apartado 5.1.4 al extremo, podemos filtrar las capturas utilizando solo el valor mínimo de cada serie, que es el que mas se debe aproximar al valor teórico, recordando que no hay valores por debajo del teórico si no se introduce un error con un ruido externo al sistema.

Los resultados se ajustan mucho al resultado esperado. En la figura 5.1.5.1 están representadas 3 medidas con este método y la medida teórica.

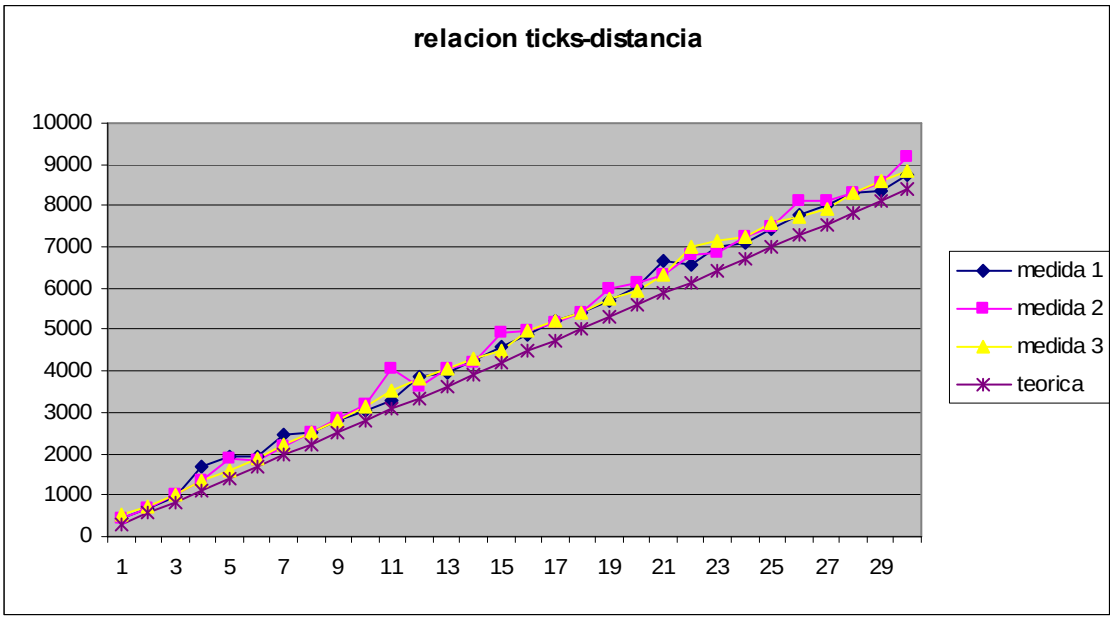


Fig. 5.1.5.1 Representación de 3 medidas utilizando los valores mínimos de cada serie de 10 valores.

Como se puede observar, las rectas que representan la relación ticks-distancia son prácticamente paralelas, con una diferencia que aumenta con la distancia y que se soluciona en el capítulo 5.2.1 donde se explica el cálculo de la distancia

El problema de este método es que al tener tanta dispersión en las medidas, podemos generar un error si no tenemos una medida mínima adecuada.

Las pruebas confirman que si capturamos 10 valores, el 90 % de las veces obtenemos un valor mínimo que se ajusta a esta gráfica. Para calcular este porcentaje, se han realizado 30 medidas, de las cuales 27 (90%) se ha calculado la distancia con menos de 5 cm. de error, y 3 (10%) se ha calculado la distancia con mas de 5 cm. de error.

5.2. Aplicación de localización

En una primera aproximación, la aplicación simplemente recogía y escribía en un fichero los datos que enviaba un solo nodo, para encontrar la relación tiempo-distancia que después se utiliza para localizar el nodo deseado.

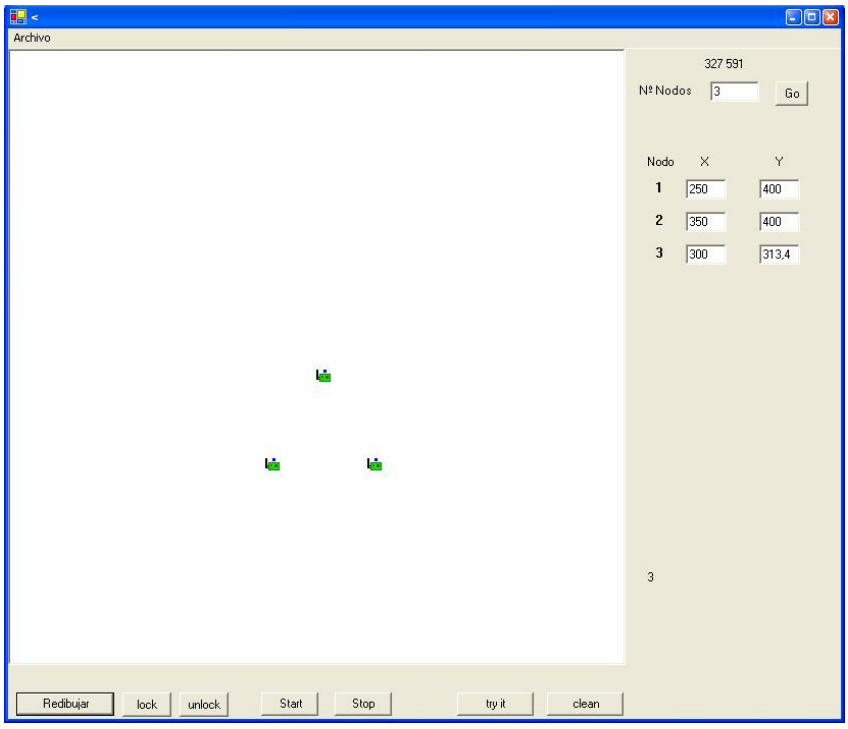
Una vez encontrada la relación, se ha aumentado la capacidad de la aplicación para poder mostrar gráficamente la situación del sistema.

En este punto se utilizan tres nodos en una posición conocida, y la aplicación captura los datos de los tres nodos.

La aplicación se encarga tanto de la parte gráfica como del procesado de las medidas para localizar el nodo.

Siguiendo los pasos, la aplicación:

- Recoge los datos enviados por los nodos de posición conocida, y guarda el valor mínimo de cada uno de ellos.
- Procesa los valores para localizar el punto donde se encuentra el nodo a localizar.
- Muestra las coordenadas del nodo a localizar.
- Muestra de forma gráfica la situación de dicho nodo.



5.2.1. Cálculo de la distancia.

La relación que se ha utilizado entre el tiempo medido en ticks de reloj y la distancia es

$$y = 29.12x + 23$$

donde “y” es el tiempo medido (en ticks de reloj) y “x” es la distancia a calcular.

$$x = \frac{y - 23}{29.12}$$

La ecuación se ha calculado a partir de los resultados de 10 medidas utilizando el mínimo de cada una de las medidas.

En la figura 5.2.1.1 se representan las funciones teórica y la utilizada por la aplicación de relación ticks-distancia.

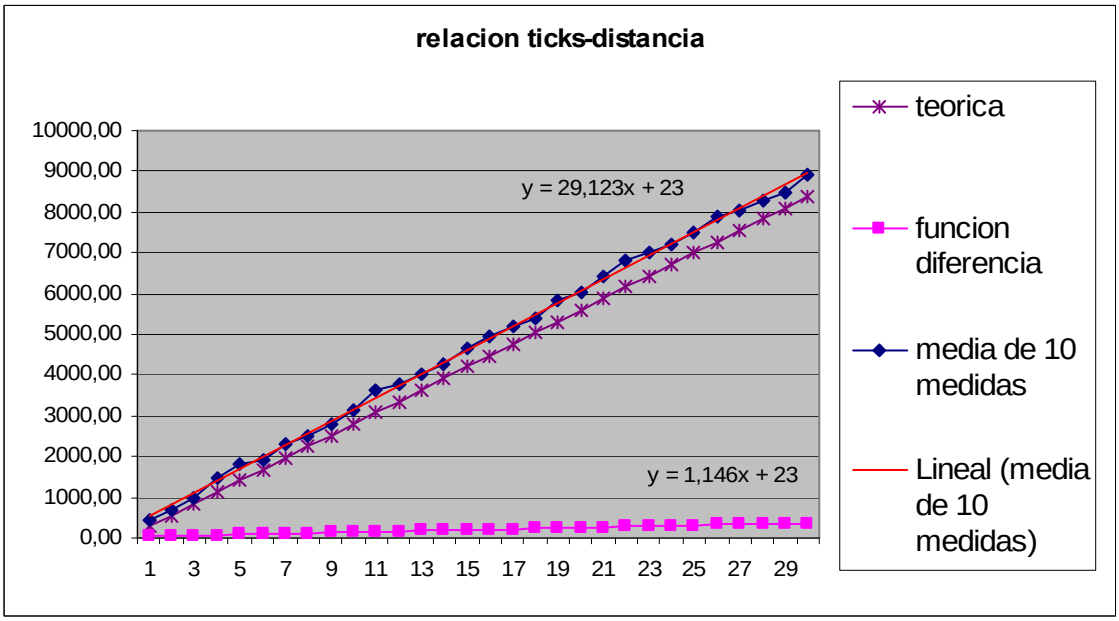


Fig. 5.2.1.1 Cálculo de la función para resolver la distancia.

La tercera línea dibujada es la diferencia con la distancia entre el valor teórico y el valor calculado.

Para calcular la distancia a partir del número de ticks se pude hacer de dos formas:

- Sustituir el número de ticks en la ecuación teórica y sumarle el error calculado con la ecuación diferencia.

- Sustituir directamente en la ecuación calculada a partir de las experiencias.

Para ahorrar el paso de calcular el valor de la ecuación diferencia, y puesto que el resultado es el mismo, la aplicación utiliza la segunda opción.

5.2.2. Posicionar el nodo

El principal obstáculo encontrado en esta parte del proyecto es que, al intentar triangular el nodo, el error que conlleva cada medida hace que las tres circunferencias representadas no coincidan en un mismo punto, incluso puede que no se corten.

En primera instancia, se utilizó el método tradicional:

Si las circunferencias se cortaban, se median los 3 puntos de corte que estuvieran mas cerca, y se calculaba el centro del triangulo resultante.

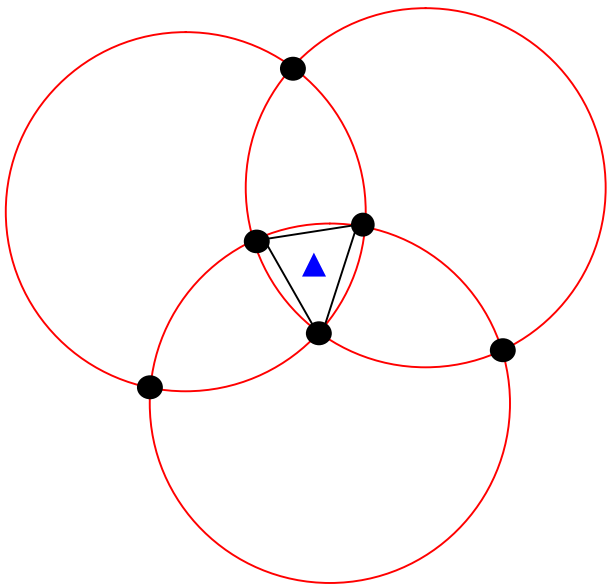


Fig. 5.2.2.1 Representación gráfica del algoritmo de localización

El problema resulta cuando las circunferencias no se cortan, ya que, aunque la medida conlleve más error, no se puede permitir que la aplicación se cuelgue, o que no devuelva ningún resultado.

Para intentar resolver el problema, se utilizó una recta común a dos circunferencias, el eje radical[2], que se basa en la propiedad de la potencia[3] de un punto respecto a una circunferencia.

[2] El eje radical de dos circunferencias no concéntricas esta formado por los puntos cuya potencia es la misma respecto a las dos circunferencias.

Si calculamos el eje radical de las circunferencias dos a dos, encontramos el centro radical de tres circunferencias (figura 5.2.2.2). De esta forma se soluciona el problema, ya que el centro radical existe tanto si las circunferencias se cortan como si no se cortan (figura 5.2.2.3).

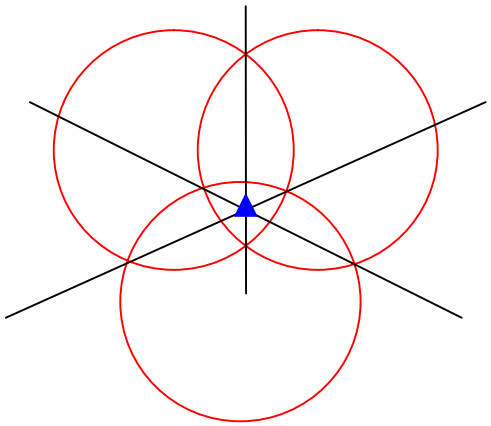


Fig. 5.2.2.2 Centro radical de 3 circunferencias que se cortan.

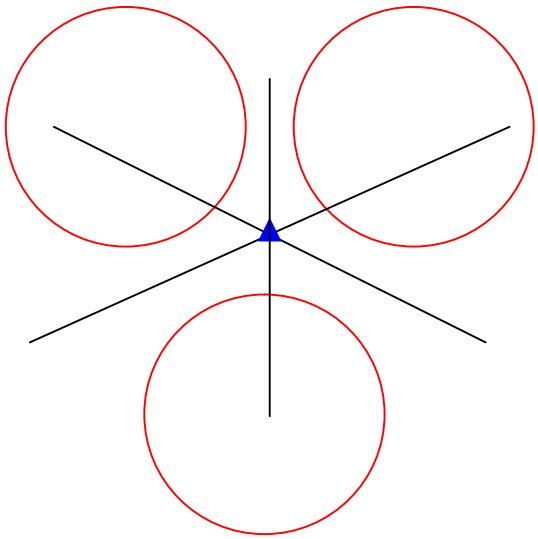
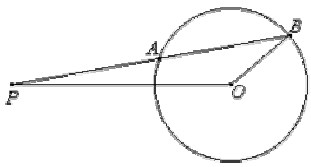


Fig. 5.2.2.3 Centro radical de 3 circunferencias que no se cortan

[3] Si desde un punto *P* trazamos una secante a una circunferencia **C** con centro *O*, que corta a ésta en los puntos *A* y *B*, el producto *PA·PB* se mantiene constante independientemente de la secante trazada. A este producto se le llama *potencia* del punto *P* respecto de la circunferencia C.



Pero después de muchas pruebas, se comprobó que el método que utiliza el eje radical añade errores al sistema en situaciones en que el otro método no lo haría.

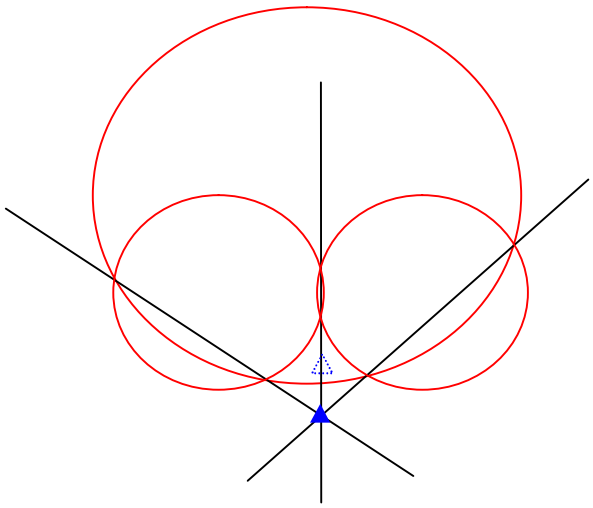


Fig. 5.2.2.3 El centro radical no siempre coincide con la localización deseada del nodo.

Si consideramos que las medidas de distancia son erróneas, o que las circunferencias no se cortan, la aproximación mediante ejes radicales es ciertamente buena.

Pero en el caso de que las medidas sean muy ajustadas, como se ha intentado en este proyecto, y que las circunferencias se corten, el método idóneo es utilizar los tres puntos mas cercanos y calcular un triangulo de incertidumbre, cuyo punto central sea la posición mas probable del nodo a localizar.

El modelo final será entonces un híbrido entre los dos métodos:

Si las tres circunferencias se cortan, utilizaremos el primer método. En el caso de que no se corten, se hará una aproximación mediante los ejes radicales de las circunferencias.

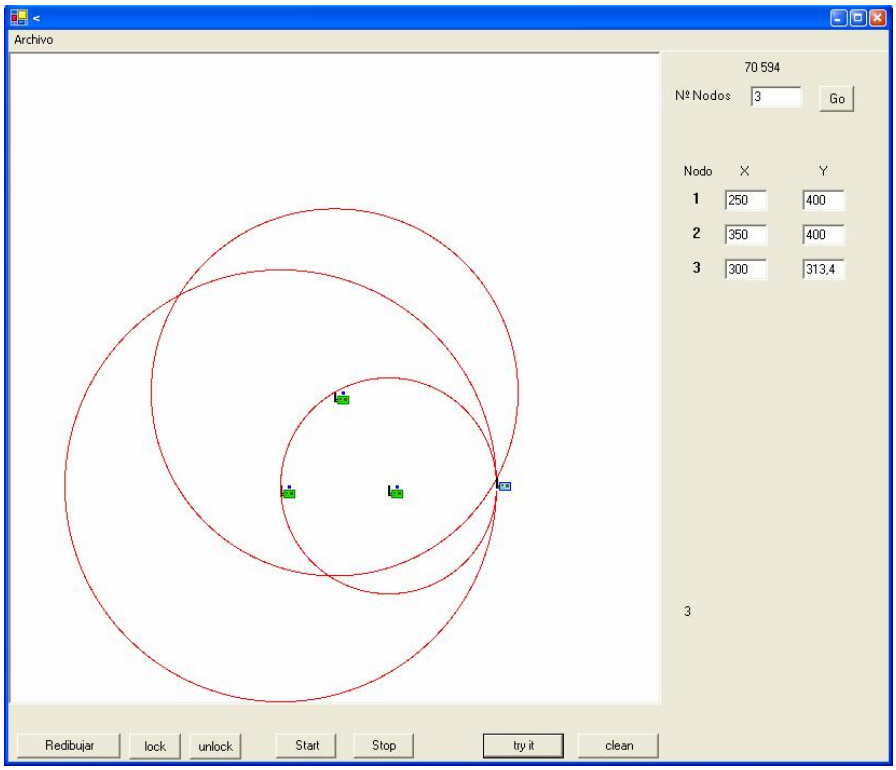


Fig. 5.2.2.4 Captura de la aplicación.

CAPITULO 6: OPTIMIZACIÓN

6.1. Concepto

El principio en que se basa la optimización es la inclusión de referencias en el sistema.

Sabiendo que el sistema implícitamente tiene errores en las medidas, podemos paliar estos errores aumentando el número de nodos que aporten datos al sistema.

Se ha reprogramado toda la aplicación para que pueda funcionar con cualquier numero de nodos de referencia.

6.2. Implementación

Para conseguir una buena aproximación al punto real, se utiliza el mismo método que antes, solo que esta vez se hace cogiendo los nodos de 3 en 3.

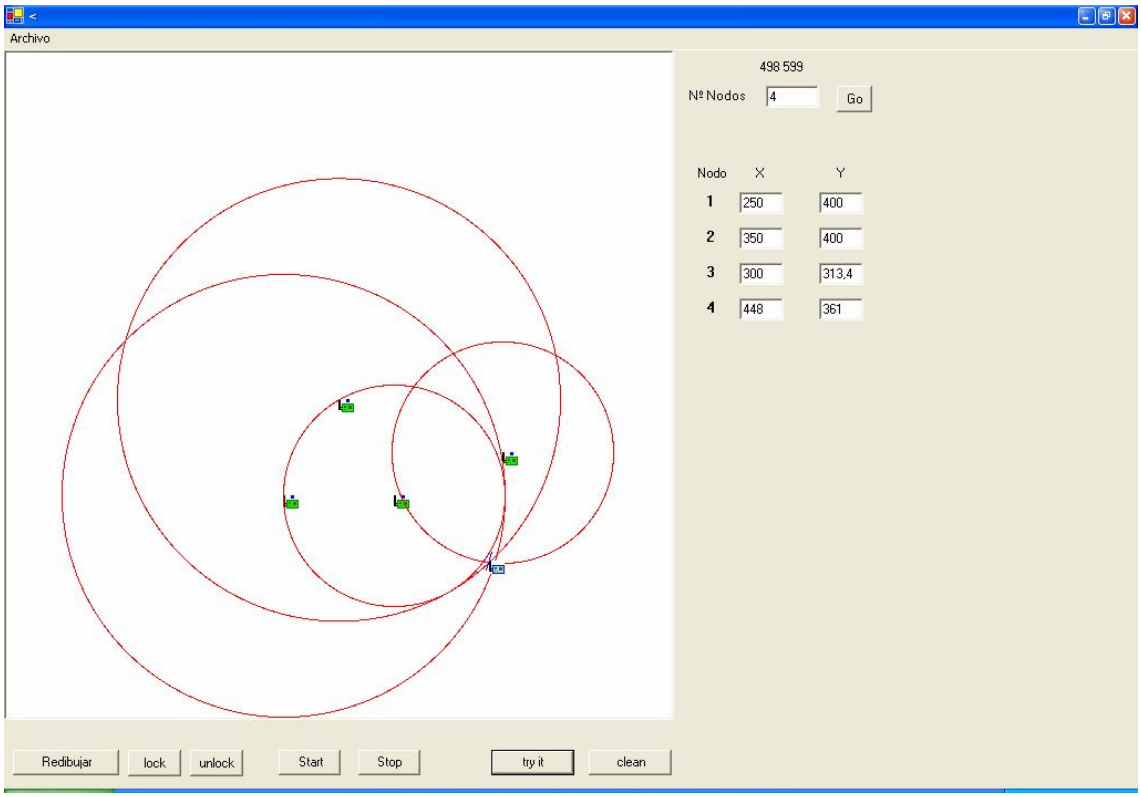


Fig. 6.1.1 Captura de la aplicación utilizando 4 nodos de referencia

Así cada tres nodos nos calcula un punto, y con todos los puntos se hace una media para calcular la posición final.

De forma análoga a la aplicación original con 3 nodos, la posición se calcula con el método tradicional de triangulación por puntos de corte o con el método del centro radical de 3 circunferencias dependiendo de si las circunferencias se cruzan o no.

Se ha probado el sistema con hasta 5 nodos de referencia, ya que uno de los micaZ del kit dejó de funcionar, se necesitan dos nodos que no sean de referencia para que el sistema funcione (el que funciona como nodo recolector y el que vamos a localizar) y el kit de CrossBow contiene 8 micaZ's.

De esta forma se ha probado la funcionalidad del sistema con el máximo número de referencias de que disponemos.

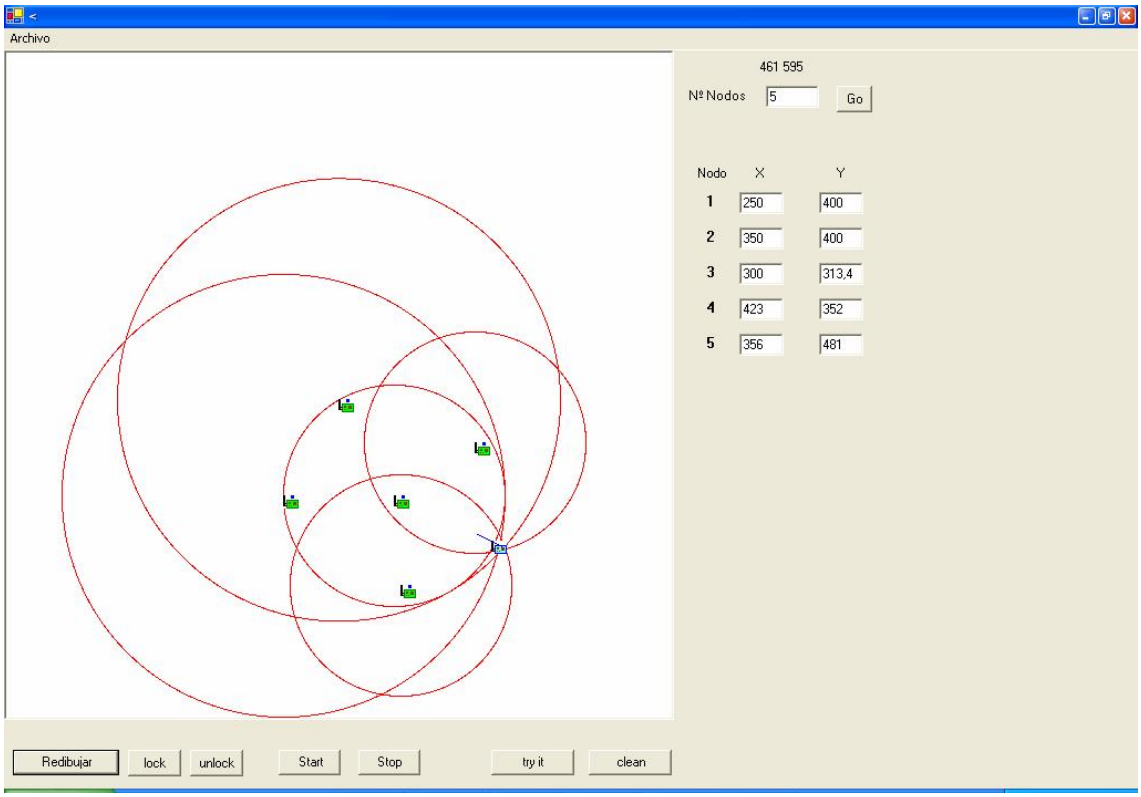


Fig. 6.1.2 Captura de la aplicación utilizando 5 nodos de referencia.

Además, una vez reprogramada la aplicación, se ha implementado una función que una vez calculado el punto final de localización, antes de mostrar el resultado de la localización, calcula la distancia media de punto medio a los puntos medidos de cada 3 circunferencias (bien sea por puntos de corte o por ejes radicales).
La función discrimina si el punto está demasiado lejos y si es así lo considera un error, despreciándolo.

Se ha comprobado que la utilización de más nodos de referencia junto con la eliminación de medidas erróneas hace más robusto el sistema, como se puede apreciar en la figura 6.1.3, donde uno de los nodos ha calculado una distancia errónea, seguramente debido a que algún ruido ha contaminado las medidas del nodo.

Pero gracias a que tenemos mas nodos en el sistema, la aplicación se ha dado cuenta de que la variación entre las distancias es demasiado grande, y ha desestimado la medida por considerarla errada.

El resultado es que el nodo se localiza en el punto correcto.

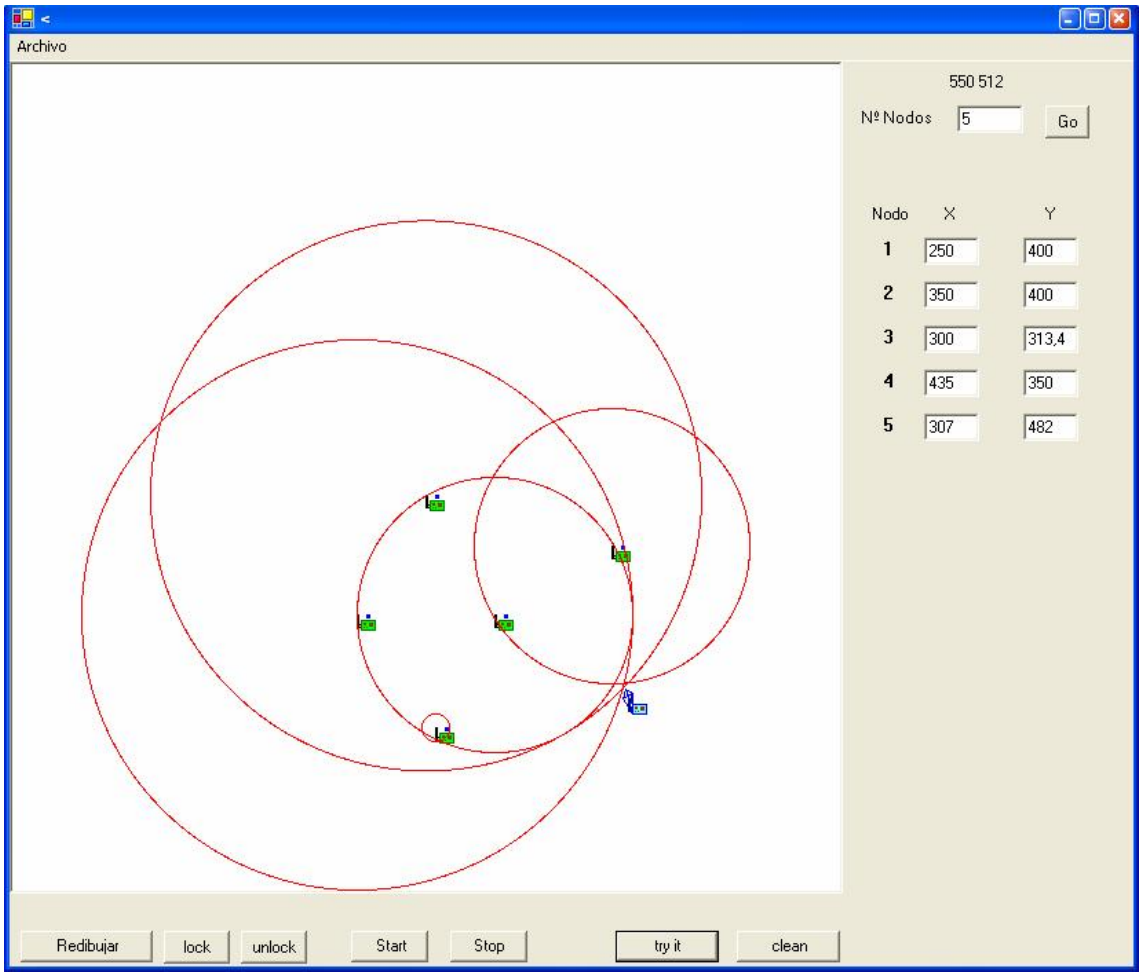


Fig. 6.1.3 captura de pantalla de la aplicación con 5 nodos de referencia donde uno de ellos muestra una medida errónea.

CAPITULO 7: CONSUMO E IMPLICACIONES MEDIOAMBIENTALES

7.1. Consumo de baterías.

Puesto que los sensores se encuentran situados, por norma general, en lugares donde no pueden ser alimentados externamente, y el cambio de batería es dificultoso, el consumo es un apartado importante a tener en cuenta a la hora de desplegar una red sensora.

En este proyecto no ha sido un punto clave el intentar un consumo mínimo de batería, pero se ha medido el consumo provocado por la aplicación.

TinyOS permite la medida instantánea del voltaje, pero en una serie de 10 medidas el consumo no es apreciable, así que se han hecho 300 medidas para poder apreciar el consumo.

Los resultados del test se ven reflejados en este resumen:

Nodo	Tiempo de la prueba	Voltaje inicial	Voltaje final	Voltaje consumido	Tiempo de vida
Generador de tonos	450 s	2693 mV	2664 mV	29 mV	≈ 6h 30m
Receptor de tonos	450 s	2598 mV	2571 mV	27 mV	≈ 7h

El tiempo de vida se ha calculado teniendo en cuenta que el voltaje de las pilas nuevas son 3 Voltios, y suponiendo que por debajo de 1.5 Voltios el nodo no funcione.

La sorpresa es la poca duración estimada de la batería, aunque hay que tener en cuenta que se esta alimentando una placa de sensores, y se esta utilizando un altavoz y un micrófono, y la optimización de los micaZ esta orientada básicamente al modo radio, ya que ZigBee es un estándar diseñado para dispositivos empotrados.

7.2. Implicaciones medioambientales

En un principio, los sensores no son dañinos para el medio ambiente, ya que no generan residuos, y no son nocivos.

Pero hay que tener en cuenta que en la mayoría de casos, cuando al sensor se le acaba la batería, se desecha, y no es fácilmente biodegradable. La batería, con el paso del tiempo dejará escapar sus compuestos que si son dañinos.

Y si hablamos en concreto de la aplicación diseñada en este proyecto, la contaminación acústica generada puede ser un problema dependiendo del emplazamiento de la red.

CAPITULO 8: CONCLUSIONES Y LÍNEAS FUTURAS

8.1. Conclusiones

El presente proyecto ha servido para comprobar que es posible un modelo de localización de sensores participantes de una red basado en la triangulación de las distancias a ciertos nodos de referencia o de posición conocida, utilizando como método de medida el tiempo de vuelo de una señal sonora y calculando la distancia a partir de la velocidad del sonido en el aire.

La validez del método, en cambio, no significa que sea el más adecuado para una red de sensores.

Hemos comprobado que el consumo que producen las múltiples repeticiones de emisión de tonos y recepción de los mismos condena a los sensores a 7 horas de vida en el peor de los casos.

Por otra parte, la investigación de los colaboradores de CrossBow, de donde son originarios los sensores utilizados en el proyecto, se esta basando en la utilización de sensores de ultrasonidos en lugar de sonido audible. En este proyecto no se han podido utilizar dichos sensores, por problemas burocráticos, pero es muy probable que los nuevos sensores de ultrasonidos mejoren el rendimiento de la batería, minimicen los errores cometidos en las medidas de los sensores y aumenten el alcance de locacalización del sistema.

8.2 Líneas futuras.

Como continuación de este proyecto, se debería reprogramar la aplicación para que fuera capaz de localizar el nodo en 3 dimensiones en lugar de las 2 dimensiones que utiliza la actual aplicación, para asemejarse mas a un escenario real. En el proceso se tendrían que sustituir las circunferencias por esferas y considerar 4 nodos de referencia como mínimo para poder localizar a un quinto nodo.

También es una labor pendiente optimizar el código que utilizan los nodos sensores para minimizar el gasto de batería, ya que la duración estimada de la batería en este proyecto no podría darse por buena en una red de sensores.

Tambien se podría mejorar el algoritmo que utiliza la aplicación del ordenador para que se consigan resultados fiables a distancias superiores a los 2 metros.

Este proyecto se iba a realizar utilizando placas de ultrasonidos, que por problemas varios no llegaron a tiempo, así que se debería repetir la experiencia utilizando ultrasonidos en lugar de sonido audible. Seria interesante comprar la mejora que se conseguiría simplemente cambiando la señal con la que se calcula la distancia.

BIBLIOGRAFÍA

[1] Antonio R. Jiménez, Fernando Seco, Carlos Prieto y Javier Roa "Tecnologías sensoriales de localización para edificios y hogares inteligentes"

[2] David Gay y otros "The *nesC* Language: A Holistic Approach to etworked Embedded Systems"

[3] http://200.58.112.165/~fi000170/Apuntes/matematica/CONICAS/DEFAULT.HTM

[4] www.tinyos.net

[5] www.xbow.com

[6] http://www.microsoft.com/spanish/msdn/spain/default.asp

[7] http://www.matematicas.net/paraiso/materia.php?id=ap_geoanali

ANEXOS

INDICE

ANEXO 1: CÓDIGO DEL EMISOR DE TONOS

1.1. Archivo de configuración

```
configuration prueba {

// this module does not provide any interface

}

implementation

{

  components Main, pruebaM, LedsC, TimerC, GenericComm as Comm,
SysTimeC, Sounder;

  Main.StdControl -> TimerC;
  Main.StdControl -> pruebaM;


  pruebaM.Leds -> LedsC;
  pruebaM.Timer -> TimerC.Timer[unique("Timer")];
  pruebaM.Send -> Comm.SendMsg[1];
  pruebaM.CommControl -> Comm;
  pruebaM.HoraL -> SysTimeC;
  pruebaM.Sound -> Sounder;
  pruebaM.FinTono -> TimerC.Timer[unique("Timer")];

}
```

1.2. Archivo módulo

```
includes AM;

module pruebaM {
  provides {
    interface StdControl;
  }
  uses {
    interface Timer;
    interface Leds;
      interface SendMsg as Send;
      interface StdControl as CommControl;
      interface SysTime as HoraL;
      interface StdControl as Sound;
      interface Timer as FinTono;
  }
}

implementation {

      TOS_Msg msg;

      command result_t StdControl.init()
      {
```

```
            call Leds.init();
            call CommControl.init();
            call Sound.init();
      return SUCCESS;
      }

      command result_t StdControl.start()
      {
      call Timer.start(TIMER_REPEAT, 1500);
            call CommControl.start();
            return SUCCESS;
      }

      command result_t StdControl.stop()
      {
      call Timer.stop();
            call CommControl.stop();
            call Sound.stop();
            return  SUCCESS;
      }


event result_t Timer.fired()
      {
            return call Send.send(2,8,&msg);
      }
event result_t Send.sendDone(TOS_MsgPtr sent, result_t success)
      {
            call Sound.start();
            call FinTono.start(TIMER_ONE_SHOT, 100);
            call Leds.greenOn();
            return SUCCESS;
      }
event result_t FinTono.fired()
      {
            call Sound.stop();
            call Leds.greenOff();
            return SUCCESS;
      }

}
```

ANEXO 2: CÓDIGO DEL RECEPTOR DE TONOS

2.1. AM2.h

```
// Message format


/**
 * @author Jason Hill
 * @author David Gay
 * @author Philip Levis
 * @author Chris Karlof
 */
#ifndef AM_H_INCLUDED
#define AM_H_INCLUDED

enum {
  TOS_BCAST_ADDR = 0xffff,
  TOS_UART_ADDR = 0x007e,
};

//#ifndef TOS_BCAST_ADDR
//#define TOS_BCAST_ADDR  0xffff
//#endif
//#define TOS_UART_ADDR   0x007e



#ifndef DEF_TOS_AM_GROUP
#define DEF_TOS_AM_GROUP 0x7d
#endif

enum {
  TOS_DEFAULT_AM_GROUP = DEF_TOS_AM_GROUP
};

uint8_t TOS_AM_GROUP = TOS_DEFAULT_AM_GROUP;

#ifndef TOSH_DATA_LENGTH
#define TOSH_DATA_LENGTH 29
#endif

#ifndef TOSH_AM_LENGTH
#define TOSH_AM_LENGTH 1
#endif

#ifndef TINYSEC_MAC_LENGTH
#define TINYSEC_MAC_LENGTH 4
#endif

#ifndef TINYSEC_IV_LENGTH
#define TINYSEC_IV_LENGTH 4
#endif

#ifndef TINYSEC_ACK_LENGTH
#define TINYSEC_ACK_LENGTH 1
#endif
```

```
typedef struct TOS_Msg
{
  /* The following fields are transmitted/received on the radio. */
  uint8_t length;
  uint8_t fcfhi;
  uint8_t fcflo;
  uint8_t dsn;
  uint16_t destpan;
  uint16_t addr;
  uint8_t type;
  uint8_t group;
  int32_t data;

  /* The following fields are not actually transmitted or received
   * on the radio! They are used for internal accounting only.
   * The reason they are in this structure is that the AM interface
   * requires them to be part of the TOS_Msg that is passed to
   * send/receive operations.
   */
  uint8_t strength;
  uint8_t lqi;
  bool crc;
  uint8_t ack;
  uint16_t time;
} TOS_Msg;

typedef struct TinySec_Msg
{
  uint8_t invalid;
} TinySec_Msg;
#ifdef MICAZOLD
enum {
  MSG_DATA_SIZE = offsetof(struct TOS_Msg, crc) + sizeof(uint16_t), //
36 by default
  DATA_LENGTH = TOSH_DATA_LENGTH,
  LENGTH_BYTE_NUMBER = offsetof(struct TOS_Msg, length) + 1,
};
#endif
enum {
  // size of the header NOT including the length byte
  MSG_HEADER_SIZE = offsetof(struct TOS_Msg, data) - 1,
//(2+1+2+2)+(TOSHeader=5)=11
  // size of the footer
  MSG_FOOTER_SIZE = 2,
  // size of the full packet-including length byte
  MSG_DATA_SIZE    =    offsetof(struct    TOS_Msg,    strength)    +
sizeof(uint16_t), //1+7+5+29+1+2
  // size of the data length
  DATA_LENGTH = TOSH_DATA_LENGTH,
  // position of the length byte
  LENGTH_BYTE_NUMBER = offsetof(struct TOS_Msg, length) + 1,
  // size of MAC Header
  TOS_HEADER_SIZE = 5,
//TOSHeader=addr(2)+type(1)+groupid(1)+length(1)
};

typedef TOS_Msg *TOS_MsgPtr;

uint8_t TOS_MsgLength(uint8_t type)
{
```

```
#if 0
  uint8_t i;

  for (i = 0; i < MSGLEN_TABLE_SIZE; i++)
    if (msgTable[i].handler == type)
      return msgTable[i].length;
#endif

  return offsetof(TOS_Msg, crc);
}
#endif
```

2.2. IntMsgMe.h

```
typedef struct IntMsgMe {
  uint32_t value;
} IntMsgMe;

enum {
  AM_INTMSG = 4
};
```

2.3. Archivo de configuración

```
configuration prueba2 {

}

implementation
{
  components Main, prueba2M, GenericComm as Comm, MicC, SysTimeC;

  Main.StdControl -> prueba2M;


  prueba2M.Receive -> Comm.ReceiveMsg[1];
  prueba2M.Send -> Comm.SendMsg[12];
  prueba2M.CommControl -> Comm;
  prueba2M.Mic -> MicC;
  prueba2M.MicCtr -> MicC;
  prueba2M.MicInt -> MicC;
  prueba2M.SysTime -> SysTimeC;

}
```

2.4. Archivo de módulo

```
includes AM;
includes IntMsgMe;

module prueba2M {
  provides {
    interface StdControl;
  }
  uses {
      interface SendMsg as Send;
    interface ReceiveMsg as Receive;
      interface StdControl as CommControl;
```

```
      interface Mic;
      interface StdControl as MicCtr;
      interface MicInterrupt as MicInt;
      interface SysTime;
  }
}

implementation {

// declaration of the static variables of the module
      uint32_t tiempo1;
      uint32_t tiempo2;
      struct TOS_Msg msg;
      IntMsgMe *Datos = (IntMsgMe *)msg.data;

// implementation of StdControl interface

command result_t StdControl.init(){
            call CommControl.init();
            call MicCtr.init();
            call Mic.gainAdjust(128);
            return SUCCESS;
      }
 command result_t StdControl.start(){
            call CommControl.start();
            call MicCtr.start();
            return SUCCESS;
      }
 command result_t StdControl.stop() {
            call CommControl.stop();
            call MicCtr.stop();
            return SUCCESS;
      }

async event result_t MicInt.toneDetected()
{
      atomic
      {
            tiempo2 = call SysTime.getTime32();
      }
      Datos->value = tiempo2 - tiempo1;
      call Send.send(0,8,&msg);
      return SUCCESS;
}
event TOS_MsgPtr Receive.receive(TOS_MsgPtr m)
{
      if (m->addr == 2)
      {
            call MicInt.enable();
            atomic
            {
                  tiempo1 = call SysTime.getTime32();
            }
      }
      return m;
}

event result_t Send.sendDone(TOS_MsgPtr sent, result_t success)
      {
            return SUCCESS;
      }
```

```
}
```

ANEXO 3: CÓDIGO UTILIZADO PARA MEDIR EL CONSUMO DE BATERIA

3.1. Nodo emisor de tonos

3.1.1. Archivo de configuración

```
configuration prueba {

// this module does not provide any interface

}

implementation

{

  components Main, pruebaM, LedsC, TimerC, GenericComm as Comm,
SysTimeC, Sounder, VoltageC;

  Main.StdControl -> TimerC;
  Main.StdControl -> pruebaM;


  pruebaM.Leds -> LedsC;
  pruebaM.Timer -> TimerC.Timer[unique("Timer")];
  pruebaM.Send -> Comm.SendMsg[1];
  pruebaM.CommControl -> Comm;
  pruebaM.HoraL -> SysTimeC;
  pruebaM.Sound -> Sounder;
  pruebaM.FinTono -> TimerC.Timer[unique("Timer")];
  pruebaM.BattControl -> VoltageC;
  pruebaM.ADC -> VoltageC;

}
```

3.1.2. Archivo de módulo

```
includes AM;
includes IntMsgMe;
module pruebaM {
  provides {
    interface StdControl;
  }
  uses {
    interface Timer;
    interface Leds;
      interface SendMsg as Send;
      interface StdControl as CommControl;
      interface SysTime as HoraL;
      interface StdControl as Sound;
      interface Timer as FinTono;
      interface ADC;
      interface StdControl as BattControl;
  }
}
```

```
implementation {

      TOS_Msg msg;
      struct TOS_Msg msg;
      IntMsgMebatt *Datos = (IntMsgMebatt *)msg.data;

    TOSH_ADC_VOLTAGE_PORT=7;
      TOSH_ACTUAL_VOLTAGE_PORT=30;

      command result_t StdControl.init()
      {
            call BattControl.init();
            call Leds.init();
            call CommControl.init();
            call Sound.init();
      return SUCCESS;
      }

      command result_t StdControl.start()
      {
            call BattControl.start();
      call Timer.start(TIMER_REPEAT, 1500);
            call CommControl.start();
            return SUCCESS;
      }

      command result_t StdControl.stop()
      {
      call BattControl.stop();
            call Timer.stop();
            call CommControl.stop();
            call Sound.stop();
            return  SUCCESS;
      }


event result_t Timer.fired()
      {
            return call Send.send(2,8,&msg);
      }

event result_t Send.sendDone(TOS_MsgPtr sent, result_t success)
      {
            call Sound.start();
            call FinTono.start(TIMER_ONE_SHOT, 100);
            call Leds.greenOn();
            return SUCCESS;
      }

event result_t FinTono.fired()
      {
            call Sound.stop();
            call Leds.greenOff();
            call Leds.yellowOn();
            call ADC.getData();
            return SUCCESS;
      }

async event result_t ADC.dataReady(uint16_t data)
      {
```

```
                Datos->batt = data;
                Datos->v = 1000;

                call BattControl.stop();
            call BattControl.init();
                call Leds.yellowOff();


                return SUCCESS;
      }

}
```

3.2. Nodo receptor de tonos

3.2.1. Archivo de configuración

```
configuration prueba2 {

}

implementation
{
  components Main, prueba2M, LedsC, GenericComm as Comm, MicC,
SysTimeC, VoltageC;

  Main.StdControl -> prueba2M;

  prueba2M.Leds -> LedsC;
  prueba2M.Receive -> Comm.ReceiveMsg[1];
  prueba2M.Send -> Comm.SendMsg[12];
  prueba2M.CommControl -> Comm;
  prueba2M.Mic -> MicC;
  prueba2M.MicCtr -> MicC;
  prueba2M.MicInt -> MicC;
  prueba2M.SysTime -> SysTimeC;
  prueba2M.BattControl -> VoltageC;
  prueba2M.ADC -> VoltageC;
}
```

3.2.2. Archivo de módulo

```
includes AM;
includes IntMsgMe;

module prueba2M {
  provides {
    interface StdControl;
  }
  uses {
      interface Leds;
      interface SendMsg as Send;
    interface ReceiveMsg as Receive;
      interface StdControl as CommControl;
      interface Mic;
      interface StdControl as MicCtr;
      interface MicInterrupt as MicInt;
      interface SysTime;
      interface ADC;
```

```
      interface StdControl as BattControl;
  }
}

implementation {

// declaration of the static variables of the module
      uint32_t tiempo1;
      uint32_t tiempo2;
      struct TOS_Msg msg;
      IntMsgMebatt *Datos = (IntMsgMebatt *)msg.data;

// implementation of StdControl interface

command result_t StdControl.init()
      {
            call BattControl.init();
            call CommControl.init();
            call MicCtr.init();
            call Mic.gainAdjust(128);
      return SUCCESS;
      }

      command result_t StdControl.start()
      {
            call BattControl.start();
            call CommControl.start();
            call MicCtr.start();
      return SUCCESS;
      }

      command result_t StdControl.stop()
      {
            call BattControl.stop();
            call CommControl.stop();
            call MicCtr.stop();
      return SUCCESS;
      }

async event result_t MicInt.toneDetected()
{
      atomic
      {
            tiempo2 = call SysTime.getTime32();
      }
      //Datos->value = tiempo2 - tiempo1;
      //call Send.send(0,8,&msg);
      call ADC.getData();
      return SUCCESS;
}
event TOS_MsgPtr Receive.receive(TOS_MsgPtr m)
{
      if (m->addr == 2)
      {
            call MicInt.enable();
            atomic
            {
                  tiempo1 = call SysTime.getTime32();
            }
      }
      return m;
```

```
}

event result_t Send.sendDone(TOS_MsgPtr sent, result_t success)
      {
            return SUCCESS;
      }

async event result_t ADC.dataReady(uint16_t data)
      {
                  Datos->batt = data;
                  Datos->v = 1000;

                  call BattControl.stop();
            call BattControl.init();
                  call Leds.yellowOff();
                  call Send.send(0,8,&msg);

                  return SUCCESS;
      }
}
```

ANEXO 4: CÓDIGO DE LA APLICACIÓN DEL PC

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Threading;
using System.Net.Sockets;
using System.Net;
namespace pruebagrafica
{
	/// <summary>
	/// Summary description for Form1.
	/// </summary>
	public class Form1 : System.Windows.Forms.Form
	{
		private System.Windows.Forms.MainMenu mainMenu1;
		private System.Windows.Forms.MenuItem menuItem1;
		private System.Windows.Forms.MenuItem menuItem2;
		private System.Windows.Forms.MenuItem menuItem3;
		private System.Windows.Forms.PictureBox dibujo;
		private System.Windows.Forms.Button dibujar;

		int NUMERO_NODOS;
		Point TMP;
		Punto[] Posiciones;
		Punto[] Puntos_rectas;
		uint[,] Valores;
		Hashtable Puntos_referencia = new Hashtable();


		int Selected;
		bool mover=false;
		Graphics G;
		static IPAddress local = Dns.Resolve(Dns.GetHostName()).AddressList[0];
		static byte[] buffer = new byte[14];
		static byte[] init = new byte[2];
		Image I1 = Image.FromFile(".\\cuadrado.bmp");
		Image I2 = Image.FromFile(".\\cuadrado2.bmp");
		Image I3 = Image.FromFile(".\\cuadrado3.bmp");
		Image nI = Image.FromFile(".\\nocuadrado.bmp");
		Image Nodo = Image.FromFile(".\\cuadradoL.bmp");
		Thread t;
		Socket C;
		Pen lapiz = new Pen(Color.Blue,1);
		uint I;

		uint tmp;
		float[] dist;
		IPEndPoint localEndPoint = new IPEndPoint(local,9001);
		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.Button lockb;
		private System.Windows.Forms.Button unlockb;
		private System.Windows.Forms.Button Start;
		private System.Windows.Forms.Button Stop;
		private System.Windows.Forms.Label distancias;
		private System.Windows.Forms.Button clean;
		private System.Windows.Forms.MenuItem menuItem4;
		private System.Windows.Forms.Label label2;
		private System.Windows.Forms.TextBox Nnodos;
		private System.Windows.Forms.Button button1;
		private System.Windows.Forms.Label label3;
		private System.Windows.Forms.TextBox N1Y;
		private System.Windows.Forms.TextBox N1X;
		private System.Windows.Forms.Label label4;
		private System.Windows.Forms.Label label5;
		private System.Windows.Forms.Label label8;
		private System.Windows.Forms.Label label9;
		private System.Windows.Forms.TextBox N2X;
		private System.Windows.Forms.TextBox N2Y;
```

```
private System.Windows.Forms.Label label11;
private System.Windows.Forms.Label label12;
private System.Windows.Forms.TextBox N3X;
private System.Windows.Forms.TextBox N3Y;
private System.Windows.Forms.Label label14;
private System.Windows.Forms.Label label15;
private System.Windows.Forms.TextBox N4X;
private System.Windows.Forms.TextBox N4Y;
private System.Windows.Forms.Label label17;
private System.Windows.Forms.Label label18;
private System.Windows.Forms.TextBox N5X;
private System.Windows.Forms.TextBox N5Y;
private System.Windows.Forms.Label Uno;
private System.Windows.Forms.Label Dos;
private System.Windows.Forms.Label Tres;
private System.Windows.Forms.Label Cuatro;
private System.Windows.Forms.Label Cinco;
private System.Windows.Forms.Label label6;
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container components = null;

public Form1()
{
	//
	// Required for Windows Form Designer support
	//
	InitializeComponent();


	//
	// TODO: Add any constructor code after InitializeComponent call
	//

}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
	if( disposing )
	{
		if (components != null)
		{
			components.Dispose();
		}
	}
	base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
	this.mainMenu1 = new System.Windows.Forms.MainMenu();
	this.menuItem1 = new System.Windows.Forms.MenuItem();
	this.menuItem2 = new System.Windows.Forms.MenuItem();
	this.menuItem3 = new System.Windows.Forms.MenuItem();
	this.menuItem4 = new System.Windows.Forms.MenuItem();
	this.dibujo = new System.Windows.Forms.PictureBox();
	this.dibujar = new System.Windows.Forms.Button();
	this.label1 = new System.Windows.Forms.Label();
	this.lockb = new System.Windows.Forms.Button();
	this.unlockb = new System.Windows.Forms.Button();
	this.Start = new System.Windows.Forms.Button();
	this.Stop = new System.Windows.Forms.Button();
	this.distancias = new System.Windows.Forms.Label();
	this.clean = new System.Windows.Forms.Button();
	this.Nnodos = new System.Windows.Forms.TextBox();
	this.label2 = new System.Windows.Forms.Label();
	this.button1 = new System.Windows.Forms.Button();
	this.label3 = new System.Windows.Forms.Label();
```

```
            this.N1Y = new System.Windows.Forms.TextBox();
            this.N1X = new System.Windows.Forms.TextBox();
            this.label4 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.Uno = new System.Windows.Forms.Label();
            this.Dos = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.label9 = new System.Windows.Forms.Label();
            this.N2X = new System.Windows.Forms.TextBox();
            this.N2Y = new System.Windows.Forms.TextBox();
            this.Tres = new System.Windows.Forms.Label();
            this.label11 = new System.Windows.Forms.Label();
            this.label12 = new System.Windows.Forms.Label();
            this.N3X = new System.Windows.Forms.TextBox();
            this.N3Y = new System.Windows.Forms.TextBox();
            this.Cuatro = new System.Windows.Forms.Label();
            this.label14 = new System.Windows.Forms.Label();
            this.label15 = new System.Windows.Forms.Label();
            this.N4X = new System.Windows.Forms.TextBox();
            this.N4Y = new System.Windows.Forms.TextBox();
            this.Cinco = new System.Windows.Forms.Label();
            this.label17 = new System.Windows.Forms.Label();
            this.label18 = new System.Windows.Forms.Label();
            this.N5X = new System.Windows.Forms.TextBox();
            this.N5Y = new System.Windows.Forms.TextBox();
            this.label6 = new System.Windows.Forms.Label();
            this.SuspendLayout();
            //
            // mainMenu1
            //
            this.mainMenu1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {

                                                    this.menuItem1});
            //
            // menuItem1
            //
            this.menuItem1.Index = 0;
            this.menuItem1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {

                                                    this.menuItem2,

                                                    this.menuItem3,

                                                    this.menuItem4});
            this.menuItem1.Text = "Archivo";
            //
            // menuItem2
            //
            this.menuItem2.Index = 0;
            this.menuItem2.Text = "-";
            //
            // menuItem3
            //
            this.menuItem3.Index = 1;
            this.menuItem3.Text = "Salir";
            this.menuItem3.Click += new System.EventHandler(this.menuItem3_Click);
            //
            // menuItem4
            //
            this.menuItem4.Index = 2;
            this.menuItem4.Text = "";
            //
            // dibujo
            //
            this.dibujo.BackColor = System.Drawing.Color.White;
            this.dibujo.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.dibujo.Location = new System.Drawing.Point(0, 0);
            this.dibujo.Name = "dibujo";
            this.dibujo.Size = new System.Drawing.Size(605, 605);
            this.dibujo.TabIndex = 2;
            this.dibujo.TabStop = false;
            this.dibujo.MouseMove += new
System.Windows.Forms.MouseEventHandler(this.dibujo_MouseMove);
            this.dibujo.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.dibujo_MouseDown);
            //
            // dibujar
```

```
//
this.dibujar.Enabled = false;
this.dibujar.Location = new System.Drawing.Point(8, 632);
this.dibujar.Name = "dibujar";
this.dibujar.Size = new System.Drawing.Size(96, 23);
this.dibujar.TabIndex = 3;
this.dibujar.Text = "Redibujar";
this.dibujar.Click += new System.EventHandler(this.dibujar_Click_1);
//
// label1
//
this.label1.Location = new System.Drawing.Point(680, 8);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(112, 16);
this.label1.TabIndex = 4;
this.label1.Text = "label1";
//
// lockb
//
this.lockb.Enabled = false;
this.lockb.Location = new System.Drawing.Point(112, 632);
this.lockb.Name = "lockb";
this.lockb.Size = new System.Drawing.Size(48, 24);
this.lockb.TabIndex = 5;
this.lockb.Text = "lock";
this.lockb.Click += new System.EventHandler(this.lockb_Click);
//
// unlockb
//
this.unlockb.Enabled = false;
this.unlockb.Location = new System.Drawing.Point(168, 632);
this.unlockb.Name = "unlockb";
this.unlockb.Size = new System.Drawing.Size(48, 24);
this.unlockb.TabIndex = 6;
this.unlockb.Text = "unlock";
this.unlockb.Click += new System.EventHandler(this.unlockb_Click);
//
// Start
//
this.Start.Enabled = false;
this.Start.Location = new System.Drawing.Point(248, 632);
this.Start.Name = "Start";
this.Start.Size = new System.Drawing.Size(56, 23);
this.Start.TabIndex = 7;
this.Start.Text = "Start";
this.Start.Click += new System.EventHandler(this.Start_Click);
//
// Stop
//
this.Stop.Enabled = false;
this.Stop.Location = new System.Drawing.Point(320, 632);
this.Stop.Name = "Stop";
this.Stop.Size = new System.Drawing.Size(56, 23);
this.Stop.TabIndex = 8;
this.Stop.Text = "Stop";
this.Stop.Click += new System.EventHandler(this.Stop_Click);
//
// distancias
//
this.distancias.Location = new System.Drawing.Point(616, 544);
this.distancias.Name = "distancias";
this.distancias.Size = new System.Drawing.Size(192, 56);
this.distancias.TabIndex = 15;
//
// clean
//
this.clean.Enabled = false;
this.clean.Location = new System.Drawing.Point(528, 632);
this.clean.Name = "clean";
this.clean.TabIndex = 14;
this.clean.Text = "clean";
this.clean.Click += new System.EventHandler(this.clean_Click);
//
// Nnodos
//
this.Nnodos.Location = new System.Drawing.Point(688, 32);
this.Nnodos.Name = "Nnodos";
```

```
            this.Nnodos.Size = new System.Drawing.Size(48, 20);
            this.Nnodos.TabIndex = 16;
            this.Nnodos.Text = "";
            //
            // label2
            //
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label2.Location = new System.Drawing.Point(616, 32);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(64, 24);
            this.label2.TabIndex = 17;
            this.label2.Text = "N° Nodos";
            //
            // button1
            //
            this.button1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.button1.Location = new System.Drawing.Point(752, 32);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(32, 24);
            this.button1.TabIndex = 18;
            this.button1.Text = "Go";
            this.button1.Click += new System.EventHandler(this.Nnodos_Enter);
            //
            // label3
            //
            this.label3.Location = new System.Drawing.Point(624, 104);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(192, 16);
            this.label3.TabIndex = 19;
            this.label3.Text = "Nodo         X                    Y";
            this.label3.Visible = false;
            //
            // N1Y
            //
            this.N1Y.Location = new System.Drawing.Point(736, 128);
            this.N1Y.Name = "N1Y";
            this.N1Y.Size = new System.Drawing.Size(40, 20);
            this.N1Y.TabIndex = 20;
            this.N1Y.Text = "400";
            this.N1Y.Visible = false;
            //
            // N1X
            //
            this.N1X.Location = new System.Drawing.Point(664, 128);
            this.N1X.Name = "N1X";
            this.N1X.Size = new System.Drawing.Size(40, 20);
            this.N1X.TabIndex = 21;
            this.N1X.Text = "250";
            this.N1X.Visible = false;
            //
            // label4
            //
            this.label4.Location = new System.Drawing.Point(712, 136);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(24, 16);
            this.label4.TabIndex = 22;
            this.label4.Text = "cm";
            this.label4.Visible = false;
            //
            // label5
            //
            this.label5.Location = new System.Drawing.Point(776, 136);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(24, 16);
            this.label5.TabIndex = 23;
            this.label5.Text = "cm";
            this.label5.Visible = false;
            //
            // Uno
            //
            this.Uno.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Uno.Location = new System.Drawing.Point(632, 128);
            this.Uno.Name = "Uno";
            this.Uno.Size = new System.Drawing.Size(24, 16);
```

```
            this.Uno.TabIndex = 24;
            this.Uno.Text = "1";
            this.Uno.Visible = false;
            //
            // Dos
            //
            this.Dos.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Dos.Location = new System.Drawing.Point(632, 160);
            this.Dos.Name = "Dos";
            this.Dos.Size = new System.Drawing.Size(24, 16);
            this.Dos.TabIndex = 29;
            this.Dos.Text = "2";
            this.Dos.Visible = false;
            //
            // label8
            //
            this.label8.Location = new System.Drawing.Point(776, 168);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(24, 16);
            this.label8.TabIndex = 28;
            this.label8.Text = "cm";
            this.label8.Visible = false;
            //
            // label9
            //
            this.label9.Location = new System.Drawing.Point(712, 168);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(24, 16);
            this.label9.TabIndex = 27;
            this.label9.Text = "cm";
            this.label9.Visible = false;
            //
            // N2X
            //
            this.N2X.Location = new System.Drawing.Point(664, 160);
            this.N2X.Name = "N2X";
            this.N2X.Size = new System.Drawing.Size(40, 20);
            this.N2X.TabIndex = 26;
            this.N2X.Text = "350";
            this.N2X.Visible = false;
            //
            // N2Y
            //
            this.N2Y.Location = new System.Drawing.Point(736, 160);
            this.N2Y.Name = "N2Y";
            this.N2Y.Size = new System.Drawing.Size(40, 20);
            this.N2Y.TabIndex = 25;
            this.N2Y.Text = "400";
            this.N2Y.Visible = false;
            //
            // Tres
            //
            this.Tres.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Tres.Location = new System.Drawing.Point(632, 192);
            this.Tres.Name = "Tres";
            this.Tres.Size = new System.Drawing.Size(24, 16);
            this.Tres.TabIndex = 34;
            this.Tres.Text = "3";
            this.Tres.Visible = false;
            //
            // label11
            //
            this.label11.Location = new System.Drawing.Point(776, 200);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(24, 16);
            this.label11.TabIndex = 33;
            this.label11.Text = "cm";
            this.label11.Visible = false;
            //
            // label12
            //
            this.label12.Location = new System.Drawing.Point(712, 200);
            this.label12.Name = "label12";
            this.label12.Size = new System.Drawing.Size(24, 16);
            this.label12.TabIndex = 32;
```

```
            this.label12.Text = "cm";
            this.label12.Visible = false;
            //
            // N3X
            //
            this.N3X.Location = new System.Drawing.Point(664, 192);
            this.N3X.Name = "N3X";
            this.N3X.Size = new System.Drawing.Size(40, 20);
            this.N3X.TabIndex = 31;
            this.N3X.Text = "300";
            this.N3X.Visible = false;
            //
            // N3Y
            //
            this.N3Y.Location = new System.Drawing.Point(736, 192);
            this.N3Y.Name = "N3Y";
            this.N3Y.Size = new System.Drawing.Size(40, 20);
            this.N3Y.TabIndex = 30;
            this.N3Y.Text = "313,4";
            this.N3Y.Visible = false;
            //
            // Cuatro
            //
            this.Cuatro.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Cuatro.Location = new System.Drawing.Point(632, 224);
            this.Cuatro.Name = "Cuatro";
            this.Cuatro.Size = new System.Drawing.Size(24, 16);
            this.Cuatro.TabIndex = 39;
            this.Cuatro.Text = "4";
            this.Cuatro.Visible = false;
            //
            // label14
            //
            this.label14.Location = new System.Drawing.Point(776, 232);
            this.label14.Name = "label14";
            this.label14.Size = new System.Drawing.Size(24, 16);
            this.label14.TabIndex = 38;
            this.label14.Text = "cm";
            this.label14.Visible = false;
            //
            // label15
            //
            this.label15.Location = new System.Drawing.Point(712, 232);
            this.label15.Name = "label15";
            this.label15.Size = new System.Drawing.Size(24, 16);
            this.label15.TabIndex = 37;
            this.label15.Text = "cm";
            this.label15.Visible = false;
            //
            // N4X
            //
            this.N4X.Location = new System.Drawing.Point(664, 224);
            this.N4X.Name = "N4X";
            this.N4X.Size = new System.Drawing.Size(40, 20);
            this.N4X.TabIndex = 36;
            this.N4X.Text = "";
            this.N4X.Visible = false;
            //
            // N4Y
            //
            this.N4Y.Location = new System.Drawing.Point(736, 224);
            this.N4Y.Name = "N4Y";
            this.N4Y.Size = new System.Drawing.Size(40, 20);
            this.N4Y.TabIndex = 35;
            this.N4Y.Text = "";
            this.N4Y.Visible = false;
            //
            // Cinco
            //
            this.Cinco.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Cinco.Location = new System.Drawing.Point(632, 256);
            this.Cinco.Name = "Cinco";
            this.Cinco.Size = new System.Drawing.Size(24, 16);
            this.Cinco.TabIndex = 44;
            this.Cinco.Text = "5";
```

```
this.Cinco.Visible = false;
//
// label17
//
this.label17.Location = new System.Drawing.Point(776, 264);
this.label17.Name = "label17";
this.label17.Size = new System.Drawing.Size(24, 16);
this.label17.TabIndex = 43;
this.label17.Text = "cm";
this.label17.Visible = false;
//
// label18
//
this.label18.Location = new System.Drawing.Point(712, 264);
this.label18.Name = "label18";
this.label18.Size = new System.Drawing.Size(24, 16);
this.label18.TabIndex = 42;
this.label18.Text = "cm";
this.label18.Visible = false;
//
// N5X
//
this.N5X.Location = new System.Drawing.Point(664, 256);
this.N5X.Name = "N5X";
this.N5X.Size = new System.Drawing.Size(40, 20);
this.N5X.TabIndex = 41;
this.N5X.Text = "";
this.N5X.Visible = false;
//
// N5Y
//
this.N5Y.Location = new System.Drawing.Point(736, 256);
this.N5Y.Name = "N5Y";
this.N5Y.Size = new System.Drawing.Size(40, 20);
this.N5Y.TabIndex = 40;
this.N5Y.Text = "";
this.N5Y.Visible = false;
//
// label6
//
this.label6.Location = new System.Drawing.Point(616, 296);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(192, 224);
this.label6.TabIndex = 45;
//
// Form1
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(816, 658);
this.Controls.Add(this.label6);
this.Controls.Add(this.Cinco);
this.Controls.Add(this.label17);
this.Controls.Add(this.label18);
this.Controls.Add(this.N5X);
this.Controls.Add(this.N5Y);
this.Controls.Add(this.Cuatro);
this.Controls.Add(this.label14);
this.Controls.Add(this.label15);
this.Controls.Add(this.N4X);
this.Controls.Add(this.N4Y);
this.Controls.Add(this.Tres);
this.Controls.Add(this.label11);
this.Controls.Add(this.label12);
this.Controls.Add(this.N3X);
this.Controls.Add(this.N3Y);
this.Controls.Add(this.Dos);
this.Controls.Add(this.label8);
this.Controls.Add(this.label9);
this.Controls.Add(this.N2X);
this.Controls.Add(this.N2Y);
this.Controls.Add(this.Uno);
this.Controls.Add(this.label5);
this.Controls.Add(this.label4);
this.Controls.Add(this.N1X);
this.Controls.Add(this.N1Y);
this.Controls.Add(this.label3);
this.Controls.Add(this.button1);
```

```
            this.Controls.Add(this.label2);
            this.Controls.Add(this.Nnodos);
            this.Controls.Add(this.clean);
            this.Controls.Add(this.distancias);
            this.Controls.Add(this.Stop);
            this.Controls.Add(this.Start);
            this.Controls.Add(this.unlockb);
            this.Controls.Add(this.lockb);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.dibujar);
            this.Controls.Add(this.dibujo);
            this.Menu = this.mainMenu1;
            this.Name = "Form1";
            this.Text = "<";
            this.ResumeLayout(false);

        }
        #endregion


        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
                Application.Run(new Form1());
        }

        private void menuItem3_Click(object sender, System.EventArgs e)
        {
                this.Close();
        }

        private void dibujar_Click_1(object sender, System.EventArgs e)
        {
                G = dibujo.CreateGraphics();
                for (int i=0;i<NUMERO_NODOS;i++)
                {
                        try
                        {
                                if (i==0)
                                {
                                        Posiciones[i].X = Convert.ToDouble(N1X.Text);
                                        Posiciones[i].Y = Convert.ToDouble(N1Y.Text);
                                }
                                else if (i==1)
                                {
                                        Posiciones[i].X = Convert.ToDouble(N2X.Text);
                                        Posiciones[i].Y = Convert.ToDouble(N2Y.Text);
                                }
                                else if (i==2)
                                {
                                        Posiciones[i].X = Convert.ToDouble(N3X.Text);
                                        Posiciones[i].Y = Convert.ToDouble(N3Y.Text);
                                }
                                else if (i==3)
                                {
                                        Posiciones[i].X = Convert.ToDouble(N4X.Text);
                                        Posiciones[i].Y = Convert.ToDouble(N4Y.Text);
                                }
                                else if (i==4)
                                {
                                        Posiciones[i].X = Convert.ToDouble(N5X.Text);
                                        Posiciones[i].Y = Convert.ToDouble(N5Y.Text);
                                }
                        }
                        catch
                        {
                                distancias.Text="No has introducido bien alguna de las
coordenadas";
                        }
                }
                for (int i=0;i<NUMERO_NODOS;i++)
                {
                        TMP = new Point((int)Posiciones[i].X,(int)Posiciones[i].Y);
                        G.DrawImageUnscaled(I1,TMP);
```

```
		}
	}

	protected void dibujo_MouseMove(object sender, System.Windows.Forms.MouseEventArgs e)
	{
		label1.Text = e.X+" "+e.Y;
		for (int i=0;i<NUMERO_NODOS;i++)
		{
			if
(((e.X)<(Posiciones[i].X+15))&&((e.Y)<(Posiciones[i].Y+12))&&((e.X)>(Posiciones[i].X))&&((e.Y)>
(Posiciones[i].Y)))
				distancias.Text="Nodo "+(i+1);

		}
		if(mover)
		{
			TMP = new Point((int)Posiciones[Selected].X,(int)Posiciones[Selected].Y);

			G.DrawImageUnscaled(nI,TMP);
			Posiciones[Selected].X=e.X;
			Posiciones[Selected].Y=e.Y;
			for (int i=0;i<NUMERO_NODOS;i++)
			{
				TMP = new Point((int)Posiciones[i].X,(int)Posiciones[i].Y);
				G.DrawImageUnscaled(I1,TMP);
			}
			if (Selected==0)
			{
				N1X.Text=Posiciones[Selected].X.ToString();
				N1Y.Text=Posiciones[Selected].Y.ToString();
			}
			else if (Selected==1)
			{
				N2X.Text=Posiciones[Selected].X.ToString();
				N2Y.Text=Posiciones[Selected].Y.ToString();
			}
			else if (Selected==2)
			{
				N3X.Text=Posiciones[Selected].X.ToString();
				N3Y.Text=Posiciones[Selected].Y.ToString();
			}
			else if (Selected==3)
			{
				N4X.Text=Posiciones[Selected].X.ToString();
				N4Y.Text=Posiciones[Selected].Y.ToString();
			}
			else if (Selected==4)
			{
				N5X.Text=Posiciones[Selected].X.ToString();
				N5Y.Text=Posiciones[Selected].Y.ToString();
			}
		}
	}

	private void dibujo_MouseDown(object sender, System.Windows.Forms.MouseEventArgs e)
	{
		bool flag_encontrado=false;
		mover = !mover;
		for (int i=0;i<NUMERO_NODOS;i++)
		{
			if
(((e.X)<(Posiciones[i].X+15))&&((e.Y)<(Posiciones[i].Y+12))&&((e.X)>(Posiciones[i].X))&&((e.Y)>
(Posiciones[i].Y)))
			{
				mover = true;
				Selected = i;
				flag_encontrado=true;
			}
			else if (flag_encontrado==false)
			{
				mover=false;
			}
		}
	}

	private void lockb_Click(object sender, System.EventArgs e)
	{
```

```
            this.dibujo.MouseDown -= new
System.Windows.Forms.MouseEventHandler(this.dibujo_MouseDown);
        }


        private void unlockb_Click(object sender, System.EventArgs e)
        {
            this.dibujo.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.dibujo_MouseDown);
        }



        private void Start_Click(object sender, System.EventArgs e)
        {
            t = new Thread(new ThreadStart(captura));
            t.Start();
        }

        private void Stop_Click(object sender, System.EventArgs e)
        {
            t.Interrupt();
            t.Abort();
            C.Close();
        }
        private void clean_Click(object sender, System.EventArgs e)
        {
            G.Clear(Color.White);
        }


        public void captura()
        {
            Valores = new uint[NUMERO_NODOS,30];
            C = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
            int[] indexes = new int[NUMERO_NODOS];

            for (int i=0;i<NUMERO_NODOS;i++)
            {
                indexes[i] = 0;
            }
            try
            {
                C.Connect(localEndPoint);


                C.Receive(init);
                foreach (byte a in init)
                {
                    Console.Write(a);
                }
                Console.Write("\n");
                C.Send(init);
                bool flag_distancias_recogidas=false;
                while (!flag_distancias_recogidas)
                {
                    C.Receive(buffer);

                    foreach (byte a in buffer)
                    {
                        Console.Write(a+"-");
                    }
                    Console.Write("\n");


                    if (buffer[0]==0)
                    {
                        I = System.BitConverter.ToUInt32(buffer,5);

                        for (int i=0; i<NUMERO_NODOS;i++)
                        {
                            if (buffer[2]==(12+i))
                            {
                                Valores[i,indexes[i]] = I;
                                indexes[i]++;
                            }
                        }
```

```
                        }
                        for (int i=0;i<NUMERO_NODOS;i++)
                        {
                                if (indexes[i]>=10)
                                        flag_distancias_recogidas=true;
                                else
                                        flag_distancias_recogidas=false;
                        }
                }
                C.Close();



                for (int nodo=0;nodo<NUMERO_NODOS;nodo++)
                {
                        tmp = Valores[nodo,0];
                        for (int valor=1;valor<indexes[nodo];valor++)
                        {
                                if (Valores[nodo,valor]<tmp)
                                        tmp=Valores[nodo,valor];
                        }
                        dist[nodo]=(tmp-23)/29;//fucnion utilizada para calcular la
distancia


                }



        //*************************************************************************
                //ya tenemos las distancias


                        Punto[] Puntos=new Punto[NUMERO_NODOS];
                Puntos_rectas = new Punto[NUMERO_NODOS*100];
                circunferencia[] circunferencias = new circunferencia[NUMERO_NODOS];
                for (int i=0;i<NUMERO_NODOS;i++)
                {
                        Puntos[i] = new Punto(Posiciones[i].X,Posiciones[i].Y);
                        circunferencias[i] = new circunferencia(Puntos[i],dist[i]);//centro y
radio
                }

                double M = 0.5;
                Bitmap bm=new Bitmap(1,1);
                bm.SetPixel(0,0,Color.Red);
                Point P;



                for (int x=0; x<=600;x++)
                {
                        for (int y=0; y<=600; y++)
                        {
                                for (int pos=0;pos<NUMERO_NODOS;pos++)
                                {
                                        if ((dist[pos]-M)<Math.Sqrt(Math.Pow(x-
Posiciones[pos].X,2) + Math.Pow(y-Posiciones[pos].Y,2)) && (dist[pos]+M)>Math.Sqrt(Math.Pow(x-
Posiciones[pos].X,2) + Math.Pow(y-Posiciones[pos].Y,2)))
                                        {
                                                P = new Point(x,y);

                                                G.DrawImageUnscaled(bm,P.X,P.Y);

                                        }
                                }
                        }
                }



                for (int i=0;i<NUMERO_NODOS;i++)
                {
```

```
                        circunferencias[i] = new
circunferencia(Puntos[i],dist[i]);//centro y radio
                    }
                    bool todas_se_cruzan=true;
                    for (int a=0;a<NUMERO_NODOS;a++)
                    {
                        for (int b=0;b<NUMERO_NODOS;b++)
                        {
                            if (!(a==b))
                            {
                                if
(circunferencias[a].Radio>=Math.Sqrt(Math.Pow((circunferencias[a].Centro.X-
circunferencias[b].Centro.X),2)+Math.Pow((circunferencias[a].Centro.Y-
circunferencias[b].Centro.Y),2)))
                                {
                                    if
(!(circunferencias[b].Radio>(circunferencias[a].Radio-
(Math.Sqrt(Math.Pow((circunferencias[a].Centro.X-
circunferencias[b].Centro.X),2)+Math.Pow((circunferencias[a].Centro.Y-
circunferencias[b].Centro.Y),2))))))
                                        todas_se_cruzan=false;
                                }
                                else if
(circunferencias[b].Radio>=Math.Sqrt(Math.Pow((circunferencias[a].Centro.X-
circunferencias[b].Centro.X),2)+Math.Pow((circunferencias[a].Centro.Y-
circunferencias[b].Centro.Y),2)))
                                {
                                    if
(!(circunferencias[a].Radio>(circunferencias[b].Radio-
(Math.Sqrt(Math.Pow((circunferencias[a].Centro.X-
circunferencias[b].Centro.X),2)+Math.Pow((circunferencias[a].Centro.Y-
circunferencias[b].Centro.Y),2))))))
                                        todas_se_cruzan=false;
                                }
                                else if
(!(circunferencias[a].Radio+circunferencias[b].Radio>=(Math.Sqrt(Math.Pow((circunferencias[a].C
entro.X-circunferencias[b].Centro.X),2)+Math.Pow((circunferencias[a].Centro.Y-
circunferencias[b].Centro.Y),2)))))
                                    todas_se_cruzan=false;
                                Console.WriteLine(todas_se_cruzan);
                            }

                        }
                    }

                    if (todas_se_cruzan==true)

                    {

                        int index_puntos=0;
                        recta tmp;
                        bool unpunto;
                        Punto[] p = new Punto[8];

                        double ycuadr,y,ind,A,B;
                        for (int a=0;a<NUMERO_NODOS;a++)
                        {
                            for (int b=0;b<NUMERO_NODOS;b++)
                            {
                                for (int c=0;c<NUMERO_NODOS;c++)
                                {
                                    if (!((b==c)||(a==c)||(a==b)))//si no es el
mismo nodo
                                    {
                                        for (int indi =0; indi<8;indi++)
                                        {
                                            p[indi]=new Punto(0,0);
                                        }
                                        unpunto=false;
                                        tmp =
interseccion_circ(circunferencias[a],circunferencias[b]);
                                        B=-tmp.b/tmp.a;
                                        A=-tmp.c/tmp.a;
                                        ycuadr = Math.Pow(B,2)+1;
                                        y = (2*A*B)-
(2*B*circunferencias[a].Centro.X)-(2*circunferencias[a].Centro.Y);
```

```
                                                        ind = Math.Pow(A,2)-
(2*A*circunferencias[a].Centro.X)+Math.Pow(circunferencias[a].Centro.X,2)+Math.Pow(circunferenc
ias[a].Centro.Y,2)-Math.Pow(circunferencias[a].Radio,2);

                                                        if ((Math.Pow(y,2)-
(4*ycuadr*ind))==0)
                                                        {

                                                                p[0].Y =(-y)/(2*ycuadr);
                                                                p[0].X = (-tmp.c-
(tmp.b*p[0].Y))/tmp.a;
                                                                unpunto=true;
                                                        }
                                                        else
                                                        {
                                                                p[0].Y =((-
y)+Math.Sqrt(Math.Pow(y,2)-(4*ycuadr*ind)))/(2*ycuadr);
                                                                p[0].X = (-tmp.c-
(tmp.b*p[0].Y))/tmp.a;

                                                                p[1].Y =((-y)-
Math.Sqrt(Math.Pow(y,2)-(4*ycuadr*ind)))/(2*ycuadr);
                                                                p[1].X = (-tmp.c-
(tmp.b*p[1].Y))/tmp.a;

                                                                unpunto=false;

                                                        }


                                                        tmp =
interseccion_circ(circunferencias[a],circunferencias[c]);
                                                        B=-tmp.b/tmp.a;
                                                        A=-tmp.c/tmp.a;
                                                        ycuadr = Math.Pow(B,2)+1;
                                                        y = (2*A*B)-
(2*B*circunferencias[a].Centro.X)-(2*circunferencias[a].Centro.Y);
                                                        ind = Math.Pow(A,2)-
(2*A*circunferencias[a].Centro.X)+Math.Pow(circunferencias[a].Centro.X,2)+Math.Pow(circunferenc
ias[a].Centro.Y,2)-Math.Pow(circunferencias[a].Radio,2);

                                                        if ((Math.Pow(y,2)-
(4*ycuadr*ind))==0)
                                                        {

                                                                p[2].Y =(-y)/(2*ycuadr);
                                                                p[2].X = (-tmp.c-
(tmp.b*p[2].Y))/tmp.a;
                                                        }
                                                        else
                                                        {
                                                                p[2].Y =((-
y)+Math.Sqrt(Math.Pow(y,2)-(4*ycuadr*ind)))/(2*ycuadr);
                                                                p[2].X = (-tmp.c-
(tmp.b*p[2].Y))/tmp.a;

                                                                p[3].Y =((-y)-
Math.Sqrt(Math.Pow(y,2)-(4*ycuadr*ind)))/(2*ycuadr);
                                                                p[3].X = (-tmp.c-
(tmp.b*p[3].Y))/tmp.a;
                                                        }
                                                        double distancia =
Math.Sqrt(Math.Pow((p[0].X-p[1].X),2)+Math.Pow((p[0].Y-p[1].Y),2));
                                                        for (int p1=0;p1<4;p1++)
                                                        {
                                                                for (int p2=0;p2<4;p2++)
                                                                {
                                                                        if (p1!=p2)
                                                                        {
                                                                                if
((Math.Sqrt(Math.Pow((p[p1].X-p[p2].X),2)+Math.Pow((p[p1].Y-p[p2].Y),2)))<distancia)
                                                                                {

        p[4]=p[p1];

        p[5]=p[p2];
```

```
        distancia = Math.Sqrt(Math.Pow((p[4].X-p[5].X),2)+Math.Pow((p[4].Y-p[5].Y),2));
                                                                                }
                                                                            }
                                                                        }
                                                                    }
                                                                    Console.WriteLine("puntos mas
cercanos\n"+p[4].X +" "+p[4].Y+"\n"+p[5].X +" "+p[5].Y);

                                                                    tmp =
interseccion_circ(circunferencias[c],circunferencias[b]);
                                                                    B=-tmp.b/tmp.a;
                                                                    A=-tmp.c/tmp.a;
                                                                    ycuadr = Math.Pow(B,2)+1;
                                                                    y = (2*A*B)-
(2*B*circunferencias[b].Centro.X)-(2*circunferencias[b].Centro.Y);
                                                                    ind = Math.Pow(A,2)-
(2*A*circunferencias[b].Centro.X)+Math.Pow(circunferencias[b].Centro.X,2)+Math.Pow(circunferenc
ias[b].Centro.Y,2)-Math.Pow(circunferencias[b].Radio,2);

                                                                    if ((Math.Pow(y,2)-
(4*ycuadr*ind))==0)
                                                                    {

                                                                            p[6].Y =(-y)/(2*ycuadr);
                                                                            p[6].X = (-tmp.c-
(tmp.b*p[6].Y))/tmp.a;
                                                                            unpunto=true;
                                                                    }
                                                                    else
                                                                    {
                                                                            p[6].Y =((-
y)+Math.Sqrt(Math.Pow(y,2)-(4*ycuadr*ind)))/(2*ycuadr);
                                                                            p[6].X = (-tmp.c-
(tmp.b*p[6].Y))/tmp.a;

                                                                            p[7].Y =((-y)-
Math.Sqrt(Math.Pow(y,2)-(4*ycuadr*ind)))/(2*ycuadr);
                                                                            p[7].X = (-tmp.c-
(tmp.b*p[7].Y))/tmp.a;

                                                                            unpunto=false;

                                                                    }

                                                                    if (unpunto==true)
                                                                    {

        Puntos_rectas[index_puntos]=p[6];

        Puntos_rectas[index_puntos+1]=p[4];

        Puntos_rectas[index_puntos+2]=p[5];
                                                                            index_puntos+=3;
                                                                    }
                                                                    else
                                                                    {

        Console.WriteLine("DISTANCIAS");
                                                                            Console.WriteLine( Math.Sqrt
(Math.Pow(p[4].X-p[5].X,2)+Math.Pow(p[4].Y-p[5].Y,2)) + Math.Sqrt (Math.Pow(p[4].X-
p[6].X,2)+Math.Pow(p[4].Y-p[6].Y,2)) + Math.Sqrt (Math.Pow(p[6].X-p[5].X,2)+Math.Pow(p[6].Y-
p[5].Y,2)) );
                                                                            Console.WriteLine( Math.Sqrt
(Math.Pow(p[4].X-p[5].X,2)+Math.Pow(p[4].Y-p[5].Y,2)) + Math.Sqrt (Math.Pow(p[4].X-
p[7].X,2)+Math.Pow(p[4].Y-p[7].Y,2)) + Math.Sqrt (Math.Pow(p[7].X-p[5].X,2)+Math.Pow(p[7].Y-
p[5].Y,2)) );

                                                                            if (( Math.Sqrt
(Math.Pow(p[4].X-p[5].X,2)+Math.Pow(p[4].Y-p[5].Y,2)) + Math.Sqrt (Math.Pow(p[4].X-
p[6].X,2)+Math.Pow(p[4].Y-p[6].Y,2)) + Math.Sqrt (Math.Pow(p[6].X-p[5].X,2)+Math.Pow(p[6].Y-
p[5].Y,2)) )<( Math.Sqrt (Math.Pow(p[4].X-p[5].X,2)+Math.Pow(p[4].Y-p[5].Y,2)) + Math.Sqrt
(Math.Pow(p[4].X-p[7].X,2)+Math.Pow(p[4].Y-p[7].Y,2)) + Math.Sqrt (Math.Pow(p[7].X-
p[5].X,2)+Math.Pow(p[7].Y-p[5].Y,2)) ))
                                                                            {

        Puntos_rectas[index_puntos]=p[6];
```

```
        Puntos_rectas[index_puntos+1]=p[4];

        Puntos_rectas[index_puntos+2]=p[5];
                                                                        index_puntos+=3;
                                                                }
                                                                else
                                                                {

        Puntos_rectas[index_puntos]=p[7];

        Puntos_rectas[index_puntos+1]=p[4];

        Puntos_rectas[index_puntos+2]=p[5];
                                                                        index_puntos+=3;
                                                                }
                                                        }
                                                }
                                        }
                                }

                                Console.WriteLine("puntos_rectas="+index_puntos);
                                Punto Aqui = new Punto(0,0);
                                for (int i = 0;i<index_puntos;i++)
                                {

        G.DrawImageUnscaled(bm,(int)Puntos_rectas[i].X,(int)Puntos_rectas[i].Y);
                                        Aqui.X+=Puntos_rectas[i].X;
                                        Aqui.Y+=Puntos_rectas[i].Y;
                                }
                                Aqui.X=Aqui.X/index_puntos;
                                Aqui.Y=Aqui.Y/index_puntos;
                                Console.WriteLine("Centro ="+Aqui.X+" "+Aqui.Y);
                                P= new Point((int)Aqui.X,(int)Aqui.Y);
                                G.DrawImageUnscaled(Nodo,P);
                                Point[] Poligono=new Point[index_puntos];
                                for (int i = 0;i<index_puntos;i++)
                                {
                                        Poligono[i].X=(int)Puntos_rectas[i].X;
                                        Poligono[i].Y=(int)Puntos_rectas[i].Y;
                                }
                                G.DrawPolygon(lapiz,Poligono);
                                label6.Text="Posicion estimada absoluta = ("+Aqui.X+"
"+Aqui.Y+")\n Posicion estimada relativa a nodo 1 = ("+(P.X-Posiciones[0].X)+" "+(P.Y-
Posiciones[0].Y)+")";

                        }
                        else
                        {
                                int index_puntos=0;
                                recta tmp,tmp1;
                                for (int a=0;a<NUMERO_NODOS;a++)
                                {
                                        for (int b=0;b<NUMERO_NODOS;b++)
                                        {
                                                for (int c = 0; c < NUMERO_NODOS; c++)
                                                {
                                                        if (!((b==c)||(a==c)||(a==b)))//si no es el
mismo nodo que b o a
                                                        {
                                                                tmp =
interseccion_circ(circunferencias[a],circunferencias[b]);
                                                                tmp1=
interseccion_circ(circunferencias[b],circunferencias[c]);
                                                                Puntos_rectas[index_puntos] =
interseccion_rectas(tmp,tmp1);
                                                                index_puntos++;
                                                        }
                                                }
                                        }
                                }



                                Console.WriteLine("index_puntos="+index_puntos);
```

```
                                index_puntos=eliminar_puntos_erroneos(index_puntos);

                                Punto Aqui = new Punto(0,0);
                                for (int i = 0;i<index_puntos;i++)
                                {
                                        Console.WriteLine(Puntos_rectas[i].X+"
"+Puntos_rectas[i].Y);

        G.DrawImageUnscaled(bm,(int)Puntos_rectas[i].X,(int)Puntos_rectas[i].Y);
                                        Aqui.X+=Puntos_rectas[i].X;
                                        Aqui.Y+=Puntos_rectas[i].Y;
                                }
                                Aqui.X=Aqui.X/index_puntos;
                                Aqui.Y=Aqui.Y/index_puntos;
                                Console.WriteLine("Centro ="+Aqui.X+" "+Aqui.Y);
                                P= new Point((int)Aqui.X,(int)Aqui.Y);
                                G.DrawImageUnscaled(Nodo,P);
                                Point[] Poligono=new Point[index_puntos];
                                for (int i = 0;i<index_puntos;i++)
                                {
                                        Poligono[i].X=(int)Puntos_rectas[i].X;
                                        Poligono[i].Y=(int)Puntos_rectas[i].Y;
                                }
                                G.DrawPolygon(lapiz,Poligono);
                                label6.Text="Posicion estimada absoluta = ("+Aqui.X+"
"+Aqui.Y+")\n Posicion estimada relativa a nodo 1 = ("+(P.X-Posiciones[0].X)+" "+(P.Y-
Posiciones[0].Y)+")";

                        }

                }
                catch
                {
                        Console.WriteLine("error de conexión");
                        t.Abort();
                }

        }

        public recta interseccion_circ (circunferencia A,circunferencia B)
        {
                //calculo del eje radical de las circunferencias

                double a,b,c;
                //calculos a mano, se puede demostrar que:

                a = ((-2)*A.Centro.X) - ((-2)*B.Centro.X);
                b = ((-2)*A.Centro.Y) - ((-2)*B.Centro.Y);
                c = (Math.Pow(A.Centro.X,2) + Math.Pow(A.Centro.Y,2) - Math.Pow(A.Radio,2)) -
(Math.Pow(B.Centro.X,2) + Math.Pow(B.Centro.Y,2) - Math.Pow(B.Radio,2));



                recta retorno = new recta(a,b,c);
                return retorno;
        }


        public Punto interseccion_rectas (recta A, recta B)
        {
                Punto retorno= new Punto(0,0);

                if ((A.b!=0)&&(B.b!=0))
                {
                        retorno.X= (((A.c*B.b)/A.b) - B.c) / (B.a - ((A.a*B.b)/A.b) );
                        retorno.Y= ((-A.a)*retorno.X)/A.b - (A.c/A.b);
                        return retorno;
                }
                else if (A.b==0)
                {
                        retorno.X = (-A.c)/A.a;
                        retorno.Y = (-B.c - (B.a*retorno.X))/B.b;
                }
                else if (B.b==0)
                {
                        retorno.X = (-B.c)/B.a;
                        retorno.Y = (-A.c - (A.a*retorno.X))/A.b;
```

```
                }

                return retorno;

        }
        private int eliminar_puntos_erroneos(int n)
        {
                //calculo la variacion de los puntos y elimino los que se pasen de la media;
                double X=0,Y=0,varX=0,varY=0;
                Punto[] Puntos_rectas2= new Punto[NUMERO_NODOS*100];
                int index=n;
                for (int i=0;i<n;i++)
                {
                        X+=Puntos_rectas[i].X;
                        Y+=Puntos_rectas[i].Y;
                }
                X=X/n;
                Y=Y/n;
                //este es el centro teorico sin eliminar los posibles puntos mas alejados
                for (int i=0;i<n;i++)
                {
                        varX+=Math.Abs(Puntos_rectas[i].X-X);
                        varY+=Math.Abs(Puntos_rectas[i].Y-Y);
                }
                varX=varX/n;
                varY=varY/n;
                int iprima=0;
                //esta es la variacion media tanto de X como de Y
                for (int i=0;i<n;i++)
                {
                        if (((Math.Abs(Puntos_rectas[i].X-
X)>(varX))||(Math.Abs(Puntos_rectas[i].Y-Y)>(varY)))&&(index>=3))
                        {
                                index--;
                        }
                        else
                        {
                                Puntos_rectas2[iprima] = new
Punto(Puntos_rectas[i].X,Puntos_rectas[i].Y);
                                iprima++;
                        }
                }
                for (int i=0;i<index;i++)
                {
                        Puntos_rectas[i] = Puntos_rectas2[i];
                }
                return index;
        }



        private void Nnodos_Enter(object sender, System.EventArgs e)
        {
                try
                {
                        if (Convert.ToInt32(Nnodos.Text)<3)
                        {
                                distancias.Text="No se puede localizar nada con menos de 3
referencias";
                        }
                        else
                        {

                                NUMERO_NODOS = Convert.ToInt32(Nnodos.Text);
                                Posiciones = new Punto[NUMERO_NODOS];
                                dist = new float[NUMERO_NODOS];
                                distancias.Text=NUMERO_NODOS.ToString();


                                for (int i = 0; i < NUMERO_NODOS; i++)
                                {
                                        Posiciones[i] = new Punto(1,1);
                                }
                        label3.Visible=true;
                                if (NUMERO_NODOS>=3)
                                {
```

```
                                        Uno.Visible=true;
                                        N1X.Visible=true;
                                        N1Y.Visible=true;
                                        Dos.Visible=true;
                                        N2X.Visible=true;
                                        N2Y.Visible=true;
                                        Tres.Visible=true;
                                        N3X.Visible=true;
                                        N3Y.Visible=true;
                                }
                                if (NUMERO_NODOS>=4)
                                {
                                        Cuatro.Visible=true;
                                        N4X.Visible=true;
                                        N4Y.Visible=true;
                                }
                                if (NUMERO_NODOS>=5)
                                {
                                        Cinco.Visible=true;
                                        N5X.Visible=true;
                                        N5Y.Visible=true;
                                }


                                dibujar.Enabled=true;
                                lockb.Enabled=true;
                                unlockb.Enabled=true;
                                Start.Enabled=true;
                                Stop.Enabled=true;
                                clean.Enabled=true;
                        }
                }
                catch
                {
                        distancias.Text="Debes introducir un numero entero";
                }

        }



        public class circunferencia
        {
                // (x - a)2 + (y - b)2 = r2
                public Punto Centro;
                public double Radio;
                public circunferencia (Punto C, double R)
                {
                        Centro = C;
                        Radio = R;
                }
        }
        public class recta
        {
                //ax + by + c = 0
                public double a,b,c;
                public recta (double a,double b,double c)
                {
                        this.a=a;
                        this.b=b;
                        this.c=c;
                }
        }
        public class Punto
        {
                public double X,Y;
                public Punto(double x, double y)
                {
                        X=x;
                        Y=y;
                }
        }

    }

}
```